NOKIA

Nokia 3300 Software Developer's Kit

Version 1.0

User's Guide

September, 2003

Copyright © Nokia 2003. All rights reserved.

This document is for use with the Nokia 3300 Software Developer's Kit. Reproduction, transfer, distribution or storage of part or all of the contents in this document in any form without the prior written permission of Nokia is prohibited.

Nokia, Nokia 3300, and the Nokia Connecting People logo are trademarks or registered trademarks of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc.

Nokia operates a policy of on-going development. Nokia reserves the right to make changes and improvements to any of the products described in this document without prior notice.

UNDER NO CIRCUMSTANCES SHALL NOKIA BE RESPONSIBLE FOR ANY LOSS OF DATA OR INCOME OR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES HOWSOEVER CAUSED.

THE CONTENTS OF THIS DOCUMENT ARE PROVIDED "AS IS". EXCEPT AS REQUIRED BY APPLICABLE LAW, NO WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE MADE IN RELATION TO THE ACCURACY, RELIABILITY OR CONTENTS OF THIS DOCUMENT. NOKIA RESERVES THE RIGHT TO REVISE THIS DOCUMENT OR WITHDRAW IT AT ANY TIME WITHOUT PRIOR NOTICE.

The availability of particular phone products may vary by region. Please check with the Nokia dealer nearest to you.

Visit Forum Nokia (http://www.forum.nokia.com), the site designed for developers using technologies supported by Nokia.

Content

About This User's Guide 5
Who Should Read This Guide? 5
Typographical Conventions in This Guide 5
Other Documents About This Product 6

1 What is the Nokia 3300 Software Developer's Kit? 7
Quick Tour of the Nokia 3300 SDK 8
Getting Information About the Content on the SDK 10
About the Diagnostic Window 10
About the Tracing Options 11
Setting Up the Nokia 3300 SDK 11
Starting the Nokia 3300 SDK Within an IDE 12
Getting Information About the SDK 12
Exiting Nokia 3300 SDK 12
What's Next? 12

2 Setting Up the Nokia 3300 SDK 15
Setting Up Preferences 15
Establishing General Settings 16
Setting the Display Size of the SDK Main Window 16
Selecting the Language of SDK Menus 16
Activating the Content Cache 17
Sending Sounds to a Sound Card 17
Setting Up Network Access 17
Setting Up a WAP Gateway 18
Setting Up Direct HTTP for MIDlets 18
Setting MIDP Provisioning and Execution Options 19
Setting the Provisioning Mode 19
Setting MIDlet Execution Options 19
Changing the MIDlet Security Policy 19
Setting the KVM Heapsize 20
Enable Persistent Record Store Emulation 20
Setting MIDP Tracing and Speed Options 21
Selecting Tracing Options 21
About MIDlet Speed Controls 21
Setting MIDlet Canvas Speed 22
Setting MIDlet KVM Speed 22
Changing Instance Numbering and Access to Diagnostics 22
What's Next? 23

3 Using the Nokia 3300 SDK 25
About Loading Content on the SDK 25
Opening Content on the SDK Standalone Application 26

Loading Content With File>Open ... 26
Loading Content Using the SDK Menus ... 26
Previewing and Browsing With NMIT ... 26
Previewing WML Files Within Adobe GoLive ... 27
Viewing the Content ... 27
Inputting Text and Commands From a Keyboard ... 28
Emulating Mouse Clicks With Keyboard Shortcuts ... 28
Inputting Text From the Keyboard ... 28
Working With Content Types ... 29
What Content Types Are Mapped to What File Types in the SDK? ... 29
What Content Types Can You Load Directly Using the Browser? ... 30
What Content Types Can You Reference in a COD Object? ... 32
Emulating a Multi-Media Memory Card (MMC) on the SDK ... 32
What's Next? ... 33

4 Getting Information About the Content on the SDK ... 35
Working with the Diagnostic Window ... 36
About the Traffic View ... 38
About MIDP View ... 41
About KVM Heapsize Status ... 42
About JAM Storage Status ... 42
About the Browser Source View ... 43
About the Browser History View ... 45
About the Log View ... 46
What's Next? ... 47

5 Using the SDK for Message Content Development ... 49
About MMS Messages ... 50
Creating an MMS Message Directly on the SDK ... 50
Creating and Previewing an MMS Message Using NMIT 4.0 ... 51
Creating and Sending an MMS Message With Adobe GoLive and NDS for MMS ... 52
Processing an MMS Message on the SDK ... 53
Working with Push Messages ... 54
Creating a Push Message with NMIT ... 55
Processing a Push Message on the SDK ... 56
What's Next? ... 57

6 Using the SDK for MIDlet Development ... 59
About MIDP MIDlets ... 59
About Loading MIDlets With and Without Provisioning ... 60
Running a MIDlet Without Provisioning ... 61
Provisioning a MIDlet ... 61
Provisioning Using the Network ... 62
Provisioning Using Local Files and the Command Line Interface ... 62
About JAD and JAR Validations for Provisioned MIDlets ... 63
Setting Up the SDK To Run MIDlets ... 63
Using the Nokia 3300 SDK With JBuilder ... 63
About the Nokia 3300 SDK Life Cycle in JBuilder ... 64
Creating a MIDlet With JBuilder and Running It on the SDK ... 64
Running a MIDlet and Displaying Trace Information in JBuilder ... 64

Using the SDK with JBuilder and NDS for J2ME 65
Using the Nokia 3300 SDK With Sun One Studio 65
What's Next? 65

7 Running the SDK from a Command Line 67

Accessing the Command Line Prompt 68
About the Command Line Syntax 68
About the Executables 69
About emulator.exe 69
About 3300.exe 69
About Options 69
Getting Information About the CLI Options 70
Options for Managing SDK Instances 70
Options for General Use 71
Options for Tracing Java Output 72
Options for General Management of Java Output 72
About Parameters 74
Working With a CLI Shell to Repeatedly Send Commands 75
Using Inboxes and Outboxes With MMS Messages 75

About This User's Guide

This manual describes how to use the Nokia 3300 Software Developer's Kit (Nokia 3300 SDK). Nokia 3300 SDK is a development tool that lets you preview how wireless content will look and work when it is ultimately deployed to an actual Nokia 3300 mobile handset. You can also use the Nokia 3300 SDK to develop content for other Nokia handsets with characteristics and functionality that are similar to those of the Nokia 3300 handset (Series 40).

Who Should Read This Guide?

You should read this guide if you plan to use the Nokia 3300 SDK in the process of creating:

- Web content designed to be browsed on a mobile phone, such as browsable WML and XHTML content
- Wireless messages such as MMS (multimedia message service) and SI/SL (Service Indictor/Service Load) Push messages
- MIDP MIDlets - Java applets in the form of JAD and JAR files. Typically, the MIDlets take the form of games or applications that can be downloaded from the Internet and used on a mobile phone.

Typographical Conventions in This Guide

This guide uses the following typographical conventions:

Convention	Explanation
`Courier`	Text that you enter (as opposed to system prompts and responses) • File paths • Commands • Program code
Italic	• Names of books and documents • New terminology
Bold	Names of Windows menus, commands, buttons, and icons
URL link	Active link to a URL.

Other Documents About This Product

These other documents contain additional information about the Nokia 3300 SDK:

- *Nokia 3300 SDK Release Notes*
- *Nokia 3300 SDK Installation and Configuration Guide*

You can find these documents in the installation directory of the SDK. They are also freely available at www.forum.nokia.com.

Additional documents about this product and its related technologies may also be available at www.forum.nokia.com.

1

What is the Nokia 3300 Software Developer's Kit?

The Nokia 3300 Software Developer's Kit (Nokia 3300 SDK) is a development tool that lets you preview how wireless content will look and work when it is ultimately deployed to a Nokia 3300 mobile handset. You can also use the Nokia 3300 SDK to develop content for other Nokia handsets with characteristics and functionality that are similar to that of the Nokia 3300 handset (series 40).

The Nokia 3300 SDK has an WML and XHTML color browser that lets the SDK receive content from web servers through a WAP gateway. Such content includes:

- Web content designed to be browsed on a mobile phone, such as MIDP, WML, and XHTML content
- Wireless messages such as MMS (multimedia message service) and SIC (Service Indictor, Compiled) Push messages
- MIDP MIDlets - Java applets in the form of JAD and JAR files. Typically, the MIDlets take the form of games or applications that can be downloaded from the Internet and used on a mobile phone.

MIDlet development is supported by these technologies:

- CLDC 1.0.4 (Connected Limited Device Configuration)
- MIDP 1.0 (Mobile Information Device Profile)
- JSR 120 1.0 (Java Specification Request)
- JSR 135 1.0 (Java Specification Request)

The browser runs on top of a WAP 1.2.1 stack.

Quick Tour of the Nokia 3300 SDK

You can run the SDK:

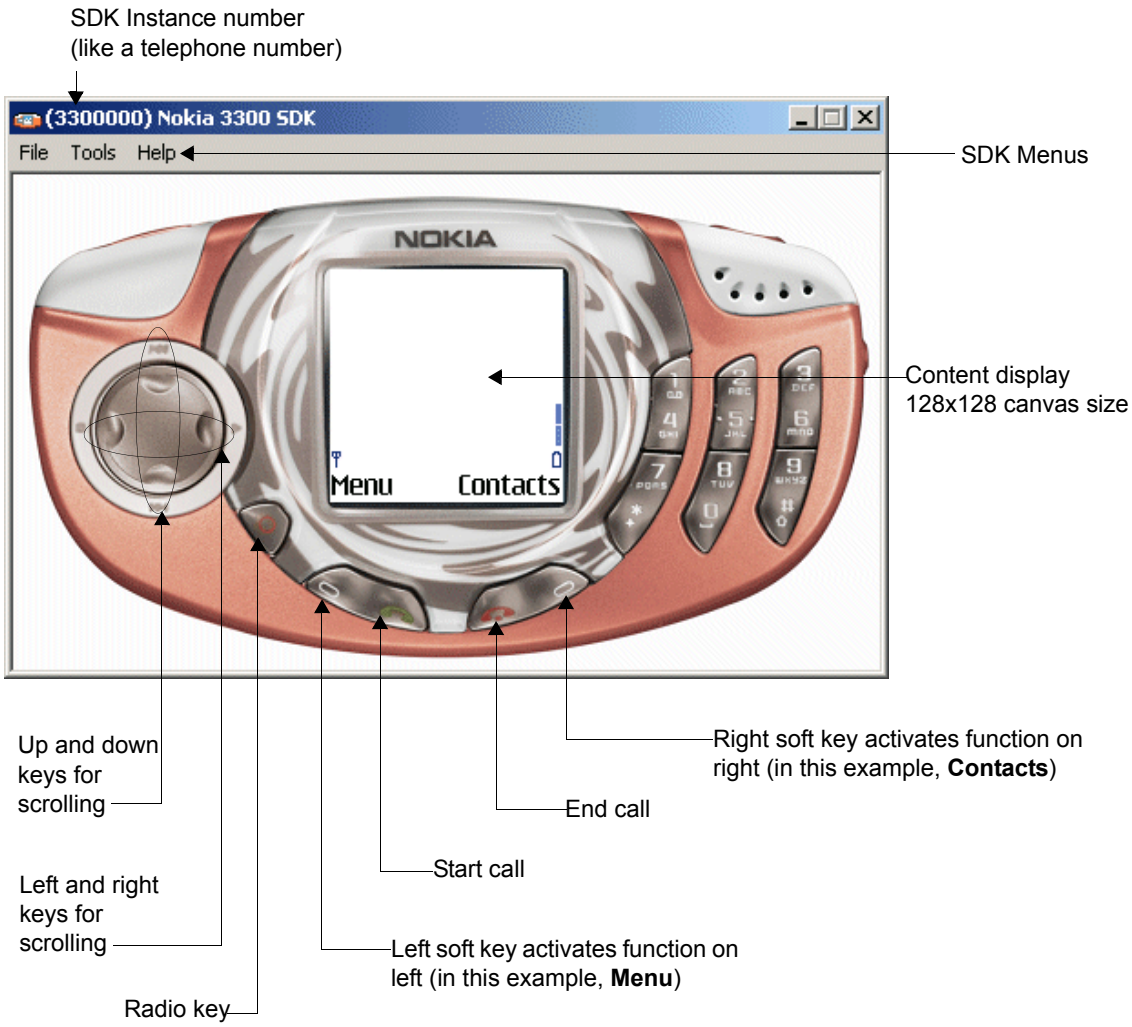
- As a standalone application by selecting **Start menu>Programs>Nokia Developer Tools>Terminal SDK>Nokia 3300 SDK 1.0>3300 SDK 1.0** or by double-clicking `<installation_directory>\bin\emulator.exe`.
- In an integrated development environment (IDE) that lets you create content with supporting applications and then immediately test the content with the Nokia 3300 SDK. (See Using the SDK for Message Content Development on page 49 or Using the SDK for MIDlet Development on page 59.)

For this quick tour, start the SDK as a standalone.

The appearance of the SDK is faithful to the appearance of the handset, although the contrast and color rendering capabilities of each might differ slightly because of differences in monitor calibration, imaging technologies, and ambient lighting.

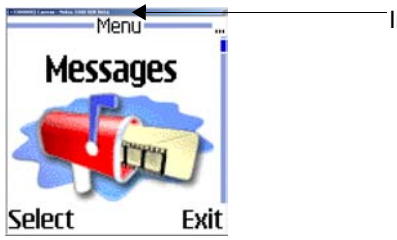
The menu structure displayed in the SDK's browser is identical to that of the Nokia 3300 phone handset, although not all menus in the SDK work identically to their Nokia 3300 handset counterpart because the SDK is not a phone but a developer's tool. For example, you cannot make a phone call on the SDK. When you place a phone call, the SDK simulates placing the call to the extent that you have to cancel the call function by clicking the End Call key, but the call is never placed.

The SDK's main window looks like this:



To simulate pressing the buttons on a Nokia 3300 mobile handset, click the corresponding keys on the SDK with the mouse. The SDK replicates the user interface of the Nokia 3300 phone handset with respect to its key definitions, menus selections, message display, backlight and vibration modes.

You can also select **Tools>Canvas** to bring up the display area of the SDK, which displays canvas of the SDK at four times the resolution (512 x 512):



Getting Information About the Content on the SDK

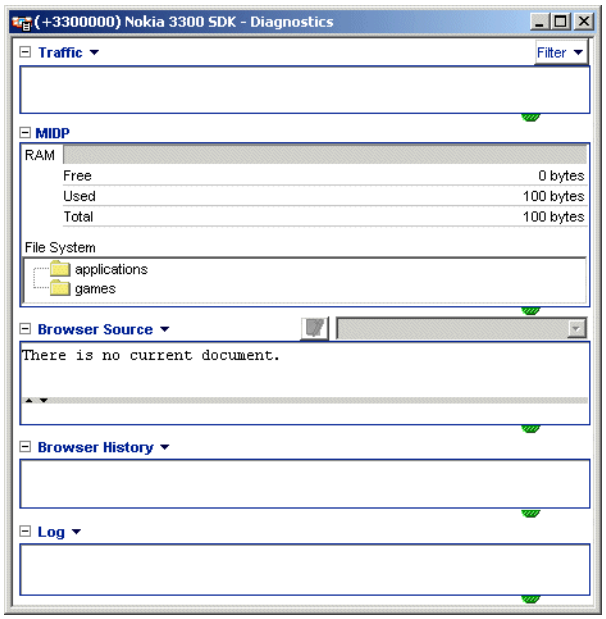
You can get information about the content on the SDK by using:

- The Diagnostic window, which provides information about how the content of the content behaves
- Command fine interface tracing options, which can help you debug a MIDP MIDlet

About the Diagnostic Window

When you select **Tools >Diagnostics**, the Diagnostics window appears and you can get information about how the content on the SDK behaves.

The Diagnostic window looks like this:



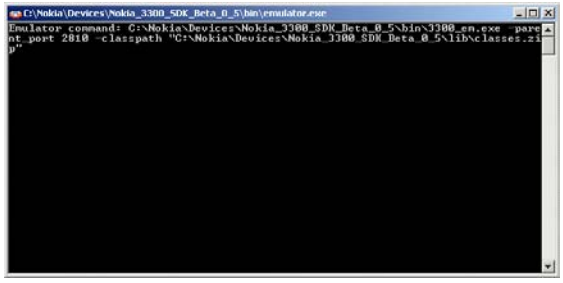
The information in the Diagnostic window lets you see information in views:

To see	Look at this view
Information about requests to and responses from the Internet	Traffic
Information about KVM and JAM status	MIDP
The source code of the content that is currently loaded on the SDK browser	Browser Source
A history list of the SDK browser	Browser History
Warning messages	Log

For more information about the Diagnostic window, see Getting Information About the Content on the SDK on page 35.

About the Tracing Options

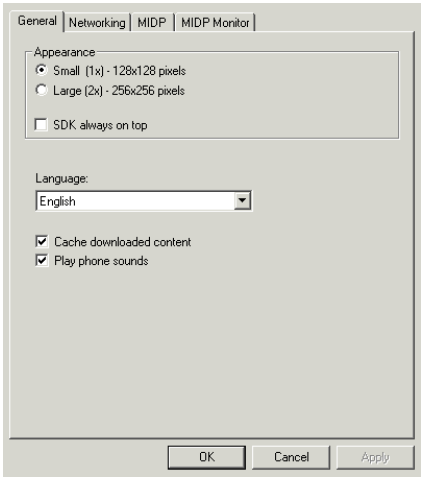
You can also activate tracing options that help you debug the content. These tracing options are displayed in the DOS Console:



For more information about tracing options and running the SDK from the command line, see Running the SDK from a Command Line on page 67.

Setting Up the Nokia 3300 SDK

To control some aspects of how the SDK operates and looks, select **Tools>Settings**. In **Settings**, you can set values for attributes such as the language in which the phone menus are displayed, the network connection mode you want to use, and settings for running MIDlets. For more information, see Setting Up the Nokia 3300 SDK on page 15. The **Settings** window looks like this:



Starting the Nokia 3300 SDK Within an IDE

While you can use the SDK as a standalone application, you can also use it within an integrated development environment (IDE) to create a powerful development and testing environment. These are some of the IDEs that support the Nokia 3300 SDK:

Browser IDEs	Messaging IDEs	MIDP Development IDEs
Nokia Mobile Internet Toolkit	Nokia Mobile Internet Toolkit	Nokia Developer's Suite for J2ME
Adobe GoLive	Nokia Developer's Suite for MMS	Borland JBuilder
	Adobe GoLive	Sun J2ME Wireless Toolkit
		Sun One Studio

You might need to configure your IDE to recognize the SDK as a supported device before you can launch the SDK from within the IDE interface. For information about configuring IDEs to work with the SDK, see the *Nokia 3300 SDK Installation and Configuration Guide* in the SDK's installation directory.

Getting Information About the SDK

You can get information about the SDK from the **Help** menu:

For information about	Select
Late-breaking news	**Help>Release Notes**
Installing and Configuring the SDK	**Help>Installation and Configuration Guide**
Using the SDK	**Help>User Guide**
This version of the SDK	**Help>About**

Exiting Nokia 3300 SDK

Select **File>Exit**. The SDK instance is terminated.

What's Next?

If you want to begin using the SDK as a standalone application right now, you can:

- Set up how the SDK operates and looks. See Setting Up the Nokia 3300 SDK on page 15.
- Begin to load content on the SDK to see how the content looks and works. See Using the Nokia 3300 SDK on page 25.

Or you might want to explore running the SDK as a standalone application from a command line. See Running the SDK from a Command Line on page 67.

2

Setting Up the Nokia 3300 SDK

You can control some aspects of how the SDK operates and looks by

- Making selections in the Preferences window (**Tools>Preferences**). See Setting Up Preferences on page 15.
- Editing the text file, `3300.user.config`. See Changing Instance Numbering and Access to Diagnostics on page 22

You can also change the default numbering style of SDK instances and turn off the Diagnostics feature (**Tools>Diagnostics**) by editing the text file, `3300.user.config`.

Setting Up Preferences

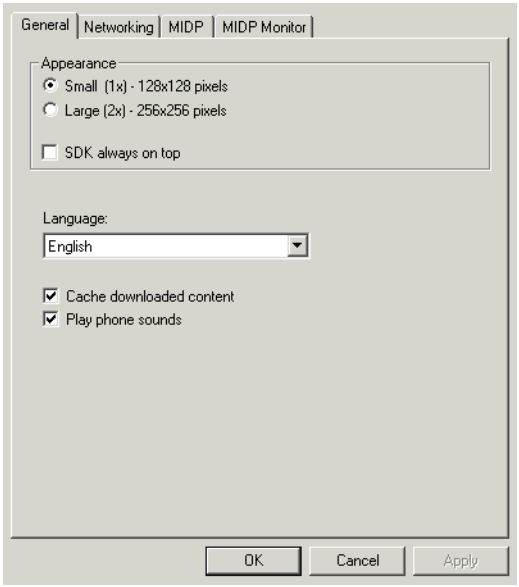
In the Preferences window (**Tools>Preferences**), you can control:

- SDK main window size, SDK menu language, cache use, and sound control
- Networking settings, such as the WAP gateway and whether to use an HTTP connection for MIDlets:

- MIDlet provisioning and execution options
- MIDlet monitoring such as tracing options, KVM and canvas speed controls

Establishing General Settings

Select **Tools>Settings** and click **General** to set up how the SDK appears and operates. This panel appears:



Setting the Display Size of the SDK Main Window

Select the image size you want for the main window of the SDK. Both sizes function identically. This setting applies only to the SDK's main window. The actual canvas size of the canvas remains constant at 128 x 128 pixels.

You can also elect to display the SDK on top of any other window on the screen by checking **SDK always on top**.

Selecting the Language of SDK Menus

For localization, you can select the language in which the menus in the SDK's browser will be displayed.This setting affects only the menus on the SDK. Internet documents are displayed in the language in which the document was created. You must restart the SDK after changing the language setting for the new setting to take effect.

These languages are automatically installed when you install the SDK: English, German, Danish, Swedish, Finnish, and Norwegian You can download other languages from www.forum.nokia.com. For more information about languages, see the *Nokia 3300 SDK Installation and Configuration Guide*.

Activating the Content Cache

The Nokia 3300 handset has a cache that stores downloaded Internet content. This feature improves browser performance. Checking **Cache downloaded content** enables the cache in the SDK, which causes the SDK to behave like the actual handset.

Unlike the handset, the SDK also provides you the option to disable this cache. Uncheck the box if you prefer to fetch a fresh copy of content over the Internet each time you browse to a location. Unchecking **Cache downloaded content** lets you observe any changes you made to the content on an Internet Web server and disables all cache directives from downloads.

Sending Sounds to a Sound Card

Check **Play phone sounds** to direct the SDK to send supported sound content to the computer's sound card. When you don't check this box, you won't hear any sound from the SDK.

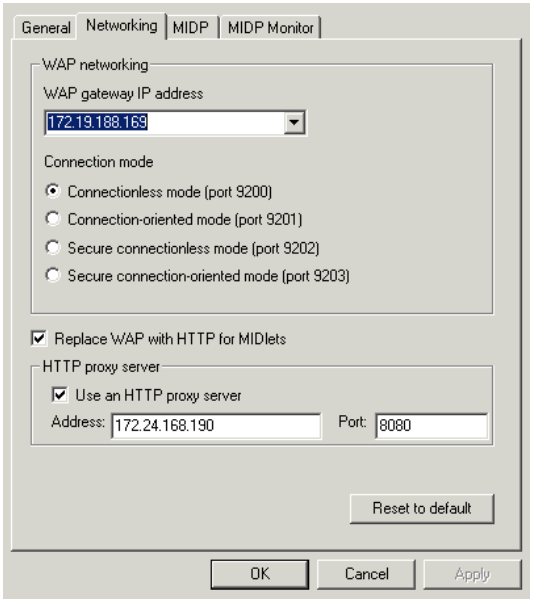
Setting Up Network Access

Both the Nokia 3300 SDK and the Nokia 3300 handset:

- Use a WAP gateway - that is, you must have a WAP gateway or gateway simulator for the SDK to communicate with the Internet
- Support MIDlets that use HTTP protocol through a WAP gateway

Unlike the handset, the SDK also lets you run MIDlets through a direct HTTP connection.When you use this option, you do not need a WAP gateway or gateway simulator. Using direct HTTP can facilitate the development of a MIDlet during the early stages. However, the communications using direct HTTP is different from communications using a WAP gateway, so you should do your final testing through a WAP gateway.

To set up network access for the SDK, select **Tools>Settings** and click **Networking** in the **Setting** dialog box. The Networking tab looks like this:



Setting Up a WAP Gateway

The SDK, like the handset, uses a WAP gateway for Internet connections. Under **WAP Networking**, enter the WAP gateway IP address and select a connection mode. This information is available from your system administrator.

Setting Up Direct HTTP for MIDlets

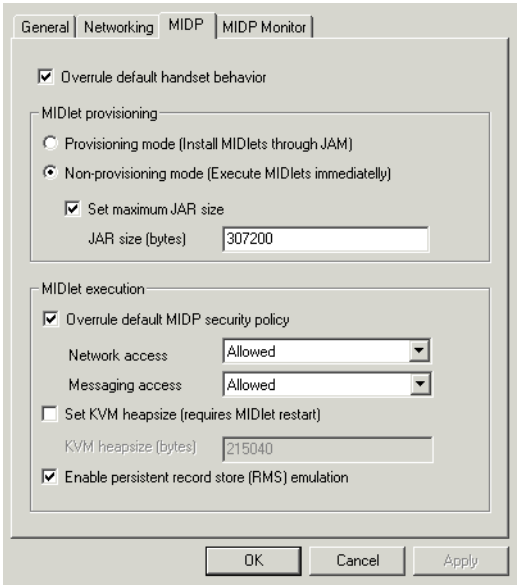
To run MIDlets with direct HTTP - that is, without requiring a WAP gateway - check **Use HTTP for MIDlets**.

You might also need to use an HTTP proxy server if the Internet requests made from your computer passes through a proxy, which would be the case, for example, if you were working within a corporate Intranet.

To set an HTTP proxy, check **Use an HTTP proxy server** and specify the address of the proxy server (for example, `113.14.176.11` or `proxy.domain`) and a port number. This information is available from your system administrator.

Setting MIDP Provisioning and Execution Options

Select **Tools>Settings** and click **MIDP** to set options that help you run MIDlets.The MIDP tab looks like this:



To access the options on the MIDP tab, check **Overrule default handset behavior**.

Setting the Provisioning Mode

For information about loading MIDlets, see About Loading MIDlets With and Without Provisioning on page 60.

In the **MIDlet Provisioning** section of this tab, you can:

- Check **Provisioning mode** to load MIDlets through JAM.
- Check **Non-provisioning mode** to run MIDlets immediately and without provisioning. (Non-provisioning is the default value.) When you select this option, you can also set the maximum JAR file size of a MIDlet the SDK can run.

Setting MIDlet Execution Options

In the **MIDlet Provisioning** section of this tab, you can configure the MIDlet execution runtime.

Changing the MIDlet Security Policy

Network security settings override any security settings you set through the SDK menus with **Applications>Games>Select Application> <MIDlet suite name>> Web Access**.

Network security settings do not apply to direct HTTP communications. Direct HTTP communications is always allowed regardless of the current settings in Network access.

Set a new security parameters by checking **Overrule default MIDP security policy** and selecting a new value for:

- **Network access** - The SDK controls a MIDlet's access to the internet.

  Allowed - The SDK automatically grants network access. (Default value on the SDK.)

  Ask first - You're asked to approve network access each time you execute a MIDlet. (Default value in the handset when MIDP 1.0 applications are provisioned.)

  Not allowed - Network access is not allowed.

- **Messaging access** - Controls access to messages

  Allowed - Accessing the message system is automatically permitted, and **Message Sent** is not displayed on the SDK display area. (Default value of the SDK.)

  Ask every time - You're asked to approve message access each time you send a message. (Default value in the handset when MIDP 1.0 applications are provisioned.)

  Not allowed - Sending messages is not allowed.

Setting the KVM Heapsize

The size of the KVM memory heap (heapsize) controls how much memory the MIDlet can take to run.

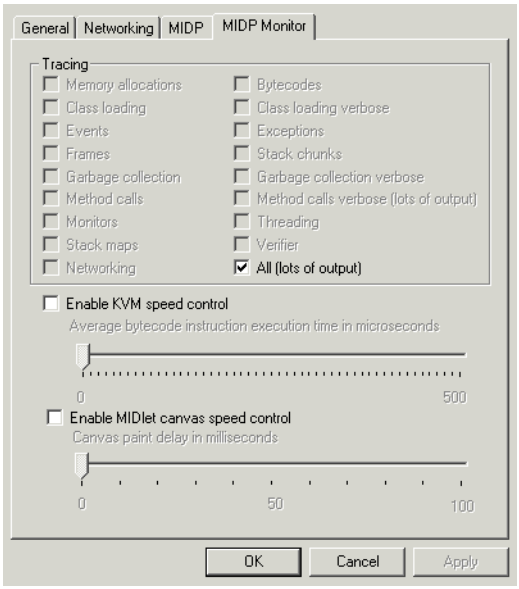
You can simulate the memory constraints of different devices. Let's say you are working on a handset that is capable of loading MIDlets up to a maximum size of 320K. If you have a MIDlet that exceeds 320K, you may want to adjust the heapsize on the SDK to verify that the MIDlet can run in a tight memory space. Set a new value for the KVM heapsize by checking **Set KVM heapsize** and specifying a new value in kilobytes by appending a k to the number or in mega-bytes by appending an M to the number. The new KVM heapsize takes effect the first time you run a MIDlet after you set the heapsize. See the `-Xheapsize` option on Options for General Use on page 71.

Enable Persistent Record Store Emulation

Enable persistent records by checking **Enable persistent record store emulation**. A persistent record is a file a MIDlet can create that remains even after you turn off the SDK. For example, a persistent record from a MIDlet might contain scores from the last time you played a game. (Default value is enabled RMS emulation.)

Setting MIDP Tracing and Speed Options

Select **Tools>Settings** and click **MIDP Monitor** to set options that let you trace the performance of a MIDlet and control the speed at which it runs. The **MIDP Monitoring** tab looks like this:



Selecting Tracing Options

Check the tracing options you want to display in the SDK's DOS Console. These tracing options trace events that occur in the KVM. (To view events that occur between the SDK and an external entity such as a gateway simulator or Internet, see About the Traffic View on page 38.)

To use tracing options within an IDE, you must set the tracing options within the IDE.

These tracing options are the same tracing options in the command line interface. For more information, see Options for Tracing Java Output on page 72.

About MIDlet Speed Controls

When you run a MIDlet on a fast computer, you might not get an accurate demonstration of how the MIDlet's runs on a handset because the simulation runs faster on the computer than it does on the handset. MIDlet speed controls (KVM and Canvas speed) let you slow down the performance of a MIDlet on a computer to better reflect the MIDlet's performance on a handset. Developing a MIDlet in an environment with a slower KVM or canvas delay speed can help you monitor and optimize your code.

However, turning on the canvas speed and KVM speed simultaneously can adversely affect the data you collect because canvas speed and KVM speed both control the speed of the SDK executable in ways that can conflict. For example, let's say you are testing a MIDlet game with an intricate graphical display that requires frequent updating, and you slow down the canvas speed to better simulate the graphical display on a handset. Simultaneously slowing down the KVM speed will additionally slow down the canvas

speed, so the performance of the MIDlet will be distorted and the data you get will be inaccurate.

For more accurate results, enable only one performance feature at a time - either the canvas speed or the KVM speed.

Setting MIDlet Canvas Speed

Use the MIDlet speed delay slider to slow down a MIDlet that displays too rapidly on a fast computer. This setting is recommended for a MIDlet with an active graphical display that requires frequent updating (for example, a game).

Move the slider to the right to increase the delay in milliseconds of the canvas paint function. Because the maximum delay you can set is 100 milliseconds, this setting has no visible effect unless the canvas repaint rate of an object is significantly greater than 10 times per second.

Setting MIDlet KVM Speed

Use this slide setting to regulate the speed of bytecode instructions in the KVM - that is, the speed at which a MIDlet runs. This setting is recommended for a MIDlet whose graphical display does not need to be updated frequently (for example, an application that calculates a mathematical equation and then displays the result).

A value of 0 (zero) indicates this feature is turned off and that the KVM is running at maximum speed. The maximum average bytecode instruction execution time is 500 microseconds.

Changing Instance Numbering and Access to Diagnostics

You can change the default numbering style of SDK instances and turn off the Diagnostics feature (**Tools>Diagnostics**) by editing the text file, `3300.user.properties`, which is located in `<installation directory>\Nokia_3300_SDK_1_0\bin\`.

Disabling the Diagnostics feature, makes the SDK start faster. The SDK Tools menu does not change when you disable the Diagnostics feature in this file; however, when you select **Tools>Diagnostics**, the Diagnostics window will not appear.

The `3300.user.properties` file looks like this:

```
#
# Nokia 3300 User Configuration File
#
#First device instance ID
DeviceFirstInstanceID = 3300000
#Diagnostics
DiagnosticsEnabled = yes
```

Change `3300000` to another number to change the default instance numbering scheme

Change `yes` to `no` to disable Diagnostics

Sometimes when you make an international call on a handset, a plus sign (+) is required for the telephone number to connect - for example, +3300000. The SDK does not require the plus sign, although the SDK distinguishes between +3300000 and 3300000, considering each a distinctly different number.

What's Next?

Now that you've customized the SDK, you can go ahead and begin:

- Displaying wireless content, as described in Using the Nokia 3300 SDK on page 25
- Developing MMS and Push messages, as described in Using the SDK for Message Content Development on page 49
- Developing MIDlets, as described in Using the SDK for MIDlet Development on page 59

Or you might want to explore running the SDK as a standalone application from a command line. See Running the SDK from a Command Line on page 67.

3

Using the Nokia 3300 SDK

Two essential tasks involved in using the SDK to develop wireless content are displaying content to check it for appearance and function and inputting commands and text in response to the content. This chapter covers the various ways you can do these tasks.

About Loading Content on the SDK

To check the appearance and function of your wireless content, you can load it on the SDK in the following ways:

- Open a local file using **File>Open**.
- Open a URL on the Internet by using **File>Open** or clicking the SDK keys to navigate to a web site - but first you might have to set up a proxy server. See Setting Up Network Access on page 17. This method is called *browsing*.

- Load content from a supporting application within an IDE. NMIT and Adobe GoLive let you display content on the SDK that you've developed with their editors. You can view the content by either clicking a button or making a menu selection in the application. This method is called *previewing*.

Opening Content on the SDK Standalone Application

You can load content on the SDK with **File>Open** or by using the menus on the SDK (as you would on the handset).

Loading Content With File>Open

To load content into the SDK when you are using the SDK as a standalone application:

1 Select **File>Open.**:

2 In the dialog box that appears, do one of the following:

- To open a local file, enter the full path of the file you want to open.
- To open mobile Internet content, enter a URL.

3 Click **OK**.

The SDK loads the content.

Loading Content Using the SDK Menus

You can also load a web site using the menus on the SDK (as you would on the handset). For example:

1 Select **Menu>Services>Go To Address.**

2 Enter a URL like: `http:\\wap.yahoo.com`.

3 Select **OK.**

The site (in this example, wap.Yahoo.com) loads on the browser.

Previewing and Browsing With NMIT

You can preview local files and browse mobile Internet content on the SDK when you use the Nokia Mobile Internet Toolkit.

To preview	In NMIT, do one of the following
Local File Content	• Select **File>Open** to open a local file in NMIT's Editor window and then click **Show** to preview the file on the SDK. • Select **File>New** to create a new file in NMIT's Editor window and then click **Show** to preview the file on the SDK.

Mobile Internet Content	Select **Tools>SDK Control Panel**. In the **SDK Control Panel** dialogs that appears: • Enter the URL you want into the address bar and then press **Return**. • Click **Bookmarks** and select a URL you have previously bookmarked.

Previewing WML Files Within Adobe GoLive

When you use Adobe GoLive, you can preview only local content on the SDK. You cannot browse mobile Internet content from within GoLive.

To preview local content:

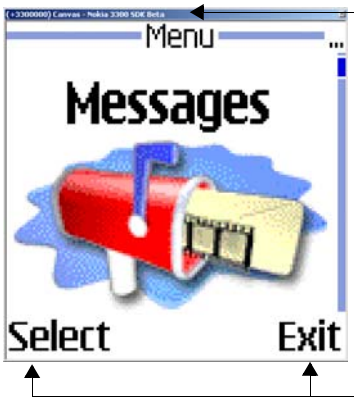
1 Do one of the following in Adobe GoLive:

- Select **File>New Special>WML Deck** to create a new WML file in GoLive's Editor window.
- Select **File>Open** to open an existing WML file in GoLive's Editor window.

2 Select **File>Preview In>3300.exe** to display it on the SDK.

Viewing the Content

Wireless content appears in the display area of the SDK's main window. To bring up the display area at four times the area of the display in the main SDK window, which is 128 by 128. The Canvas is 512 by 512. To display the Canvas, select **Tools>Canvas**. The Canvas window appears:



Instance Identifier

Click the left mouse button to select the left choice (in this example, **Select**) and the right mouse button to select the right choice (in this example, **Exit**).

In the Canvas, you also:

- select clicking the left mouse button to select the left choice and the right mouse button to select the right choice.
- Type text directly into fields.
- Scroll with the mouse scroll key.

The display in the Canvas and the phone image are synchronized. Despite the appearance of the main and Canvas windows on the desktop, the actual size of the canvas remains constant at 128 x 128 pixels.

Inputting Text and Commands From a Keyboard

In addition to clicking on the SDK GUI, you can use the computer keyboard to activate the SDK's soft keys and to enter information into the SDK.

Emulating Mouse Clicks With Keyboard Shortcuts

Here's a list of each keyboard key or key combination that performs an action on the SDK equivalent to clicking on the SDK GUI:

SDK Soft Key	Computer Keyboard Equivalent
Left soft key	F9 F11 Enter
Right soft key	F12 Backspace
Cursor Up	Up arrow
Cursor Down	Down arrow
Cursor Left	Left arrow
Cursor Right	Right arrow
Send	NumPad / (slash)
End	NumPad - (minus)
# (number sign)	NumPad . (period)
* (asterisk)	NumPad *
Numbers 0 through 9	NumPad 0 through NumPad 9
Music key	<Ctrl> + <m>

Inputting Text From the Keyboard

You can type on your computer keyboard and have the characters appear in the display screen of the SDK. For example, if the SDK browser is currently displaying a text entry box contained in an XHTML page you browsed to, you can enter characters into the box by typing on the keyboard. Characters entered into the SDK in this manner are not mapped directly to the SDK soft keys, as are the shortcut keys listed in Emulating Mouse Clicks With Keyboard Shortcuts on page 28.

For example, if you press the shortcut key Numpad 5, you will see the Number 5 soft key on the SDK flash briefly, indicating a soft key press on the SDK GUI, and the letter J appears in the text box. If you press the 5 key on the keyboard, the SDK soft key does not flash and a 5 appears in the text box because the 5 key is not mapped directly to an SDK soft key.

When the SDK is in edit mode (a small pen icon appears in the upper left corner of the display) and dictionary (that is, predictive text or T9) is not on, you can use all the alphanumeric PC keys (a-z, A-Z, 0-9), some of the non-alphanumeric characters (!@#$%^&* ()+:;' " , <.>/?[]-_={}|\~) and space key to enter text. Those characters are mapped to the corresponding non-numpad PC keyboard keys. This feature is not available for the following languages:

- Arabic
- Chinese (Simplified and Traditional)
- Hebrew
- Thai
- Vietnamese

The full keyboard input is only implemented for text boxes.

Working With Content Types

On the SDK, you can browse directly to some content types using **File>Open**. You can access other content types if they are in line with or referenced from a WML, HTML, or COD file.

What Content Types Are Mapped to What File Types in the SDK?

Because local files do not go through a server, the SDK provides an internal mapping between file types and content types. For example, when you use **File>Open** to load a local `.sic` file, the SDK uses content type `application/vnd.wap.sic`.

This table lists the mappings between file types and content types:

File suffix	Content type presented to the SDK	File Content
`.awb`	`audio/amr-wb`	Audio file

`.xhtml`
`.html`
`.htm`
	`text/html`	XHTML file

`.wml`	`text/vnd.wap.wml`	WML file
`.cod`	`text/x-co-desc`	Content Descriptor file
`.mms`	`application/vnd.wap.mms-message`	MMS message file
`.sic`	`application/vnd.wap.sic`	Encoded WAP Push SI message

File suffix	Content type presented to the SDK	File Content
.gif	image/gif	GIF file
.png	image/png	PNG file
.jpg .jpeg	image/jpeg	JPEG files
.wbmp	image/vnd.wap.wbmp	WBMP file
.mid .midi	audio/midi	MIDI sound files
.nrt .ott .rng	application/vnd.nokia.ringing-tone	Nokia Ring Tone files
.jad	text/vnd.sun.j2me.app-descriptor	MIDP Java JAD file
.jar	application/java-archive	MIDP Java JAR file
.wmlsc	application/vnd.wap.wmlsc	Compiled WMLScript
.vcs	text/x-vCalendar	Calendar file
.vcf	text/x-vCard	Card file
.bmp	image/bmp	Image file

What Content Types Can You Load Directly Using the Browser?

You can load the content types listed in this section by:

- Making an HTTP request directly through **File>Open**. For example:
  `http://www.ayg.com`
- Referencing the content type from within WML, HTML, or XHTML code to provide a link to another item such as a new web page or image (also known as an absolute reference). For example:
  `<a href=http://www.nokia.com/Click_me.wbmp>Click me</a>`

These content types are:

- `text/vnd.wap.wml`
- `text/html`
- `application/xhtml+xml`
- `text/x-vCard`
- `text/x-vCalendar`
- `text/x-co-desc`
- `image/gif`
- `image/jpeg`
- `image/png`
- `image/vnd.wap.wbmp`
- `image/bmp`

- image/vnd.nok-wallpaper
- application/vnd.nokia.ringing-tone
- audio/mid
- audio/midi
- audio/x-mid
- audio/x-midi
- audio/sp-midi
- audio/amr-wb
- text/vnd.sun.j2me.app-descriptor
- application/java
- application/java-archive
- application/x-java-archive
- application/vnd.wap.sic
- application/vnd.wap.slc
- application/x-wap-prov.browser-bookmarks
- application/x-wap-prov.browser-settings
- application/x-nokiagamedata
- image/vnd.nok-oplogo-color

WMLScript can be referenced with href only within a WML, HTML, or XHTML content. The browser cannot directly open WMLScript.

- image/vnd.wap.wbmp
- application/vnd.wap.wmlsc

In-line content types within WML, HTML, and XHTML code are displayed as a subcomponent of a page. Generally, all content types can be referenced but only images can be displayed in-line. Some exceptions apply, however. For example, wallpaper (`image/vnd.nok-wallpaper`) cannot be displayed in-line but cannot be referenced and a snapshot (`image/vnd-nok-camera-snap`) can be referenced but cannot be displayed in-line.

An example of an in-line format is: `<img src="foo.gif" alt="Image Text"/>`

The content types you can use in-line are:

- image/gif
- image/jpeg, image/jpg
- image/png
- image/vnd.wap.wbmp
- image/bmp

What Content Types Can You Reference in a COD Object?

A COD (content download object) is a Nokia-proprietary file format generated by DRM (digital rights management). A URL that you load on the SDK browser can reference a COD object that contains these content types:

- `text/x-vCard`
- `text/x-vCalendar`
- `image/gif`
- `image/jpeg`
- `image/png`
- `image/vnd.wap.wbmp`
- `image/bmp`
- `image/vnd.nok-wallpaper`
- `application/vnd.nokia.ringing-tone`
- `audio/mid`
- `audio/midi`
- `audio/x-mid`
- `audio/x-midi`
- `audio/sp-midi`
- `audio/amr-wb`
- `text/vnd.sun.j2me.app-descriptor`
- `application/java`
- `application/x-nokiagamedata`
- `image/vnd.nok-oplogo-color`

Emulating a Multi-Media Memory Card (MMC) on the SDK

The Nokia 3300 handset can use a Multi-Media Memory Card (MMC) to augment its memory. The SDK emulates an MMC card by creating a default MMC directory structure for each SDK instance. The MMC directories are located in:

`C:\<Nokia_3300_SDK_1_0 install directory>\bin\Storage\+<instance identifier>\MMC`

If the content of the memory card is empty when you use the SDK, the SDK creates a default directory structure with the following directories: Applications, Backup, Music, Recordings, Tones and Track Lists.

The MMC directory structure is persistent and applies to any SDK instance with a matching instance identifier. The SDK reads the content of MMC folder from the hard drive only at startup. A running SDK instance is not affected if you change the content of the MMC directory after the instance starts.

If the MMC directory for an SDK instance already exists, the SDK uses the directory as it exists. You can modify the MMC directory, and the SDK will display the sub-directory names as folders and let you save content in those folders through the SDK instance.

The benefit of changing the MMC directory on the PC hard drive is that you can transfer a custom directory structure to the handset if you have an MMC card reader on our computer.

What's Next?

Now that you've loaded and viewed content on the Nokia 3300 SDK, you can get information about that content in Getting Information About the Content on the SDK on page 35.

Or you can begin to develop:

- MMS and Push messages. See Using the SDK for Message Content Development on page 49.
- MIDlets. See Using the SDK for MIDlet Development on page 59.

You might also want to explore running the SDK as a standalone application from a command line. See Running the SDK from a Command Line on page 67.

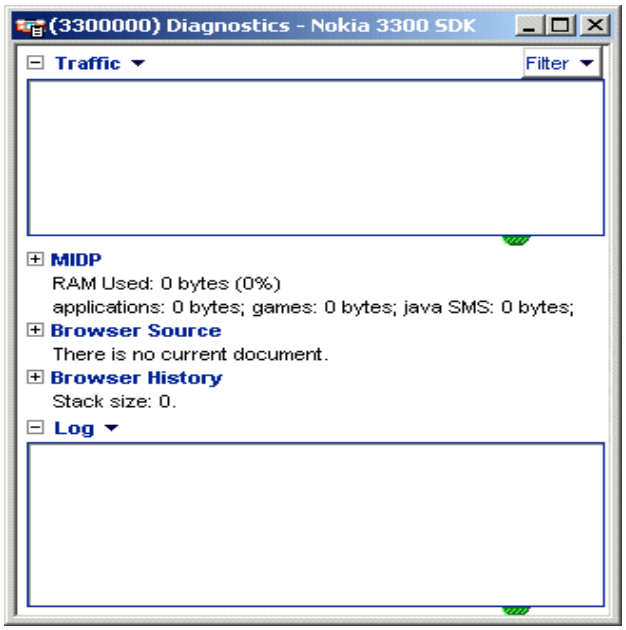
4

Getting Information About the Content on the SDK

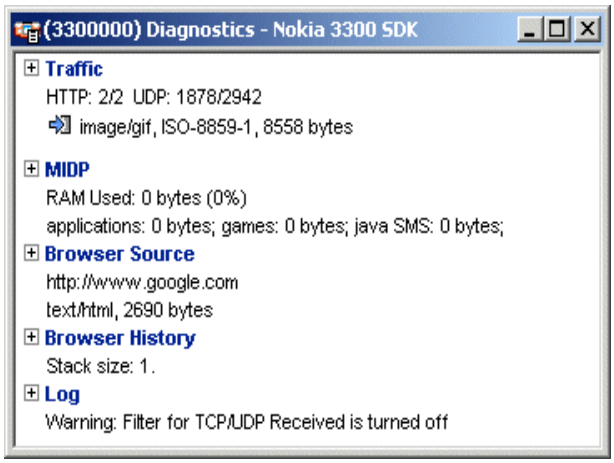
To help you analyze and troubleshoot the content on the SDK, you can view information about the current content on the SDK in the Diagnostic window. The information in this window is dynamic and changes as the content on the SDK changes.

Working with the Diagnostic Window

The Diagnostics window displays information that can help you determine whether the content on the SDK is working properly. To see the Diagnostics window, select **Tools >Diagnostics**. If nothing is loaded on the browser, the Diagnostic window looks like this when you first open it:

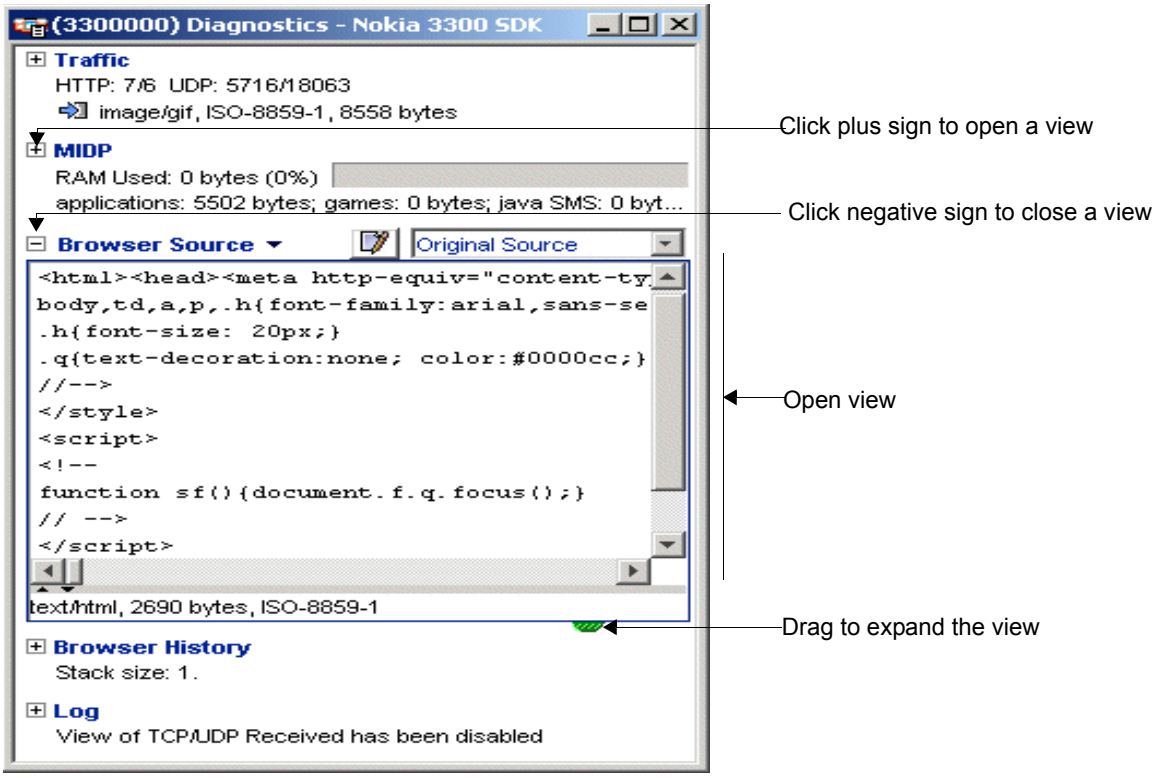


When content is loaded on the SDK and none of the views are open, the Diagnostic window looks something like this, depending on the content:



When a view in the Diagnostics window is closed, you can see a summary of the contents in each view.

You can open, close, and adjust the size of a view. With an open Browser Source view, the Diagnostic window looks something like this, depending on what's loaded on the browser:



To display only the views you want:

1 Open the view you want to see.

2 Right click that view. Select **Views** from the menu that appears to open a list of views. Uncheck the views you do not want to see.

3 The checked views expand to fill the space of the Diagnostic window.

To recreate some of the examples in this chapter:

1 Select **Tools >Diagnostics**.

2 Select **File>Open**.

3 Enter `http://www.spruce.jp/freemidlets/` in the text field and wait until the web content loads on the browser.

The Diagnostic window moves with the main window whenever you move the main window, but you can move the Diagnostics window independently of the main window. Changes you make to the Diagnostics window are persistent between instances.

The Diagnostics window displays these different categories of information:

To see	See this section
Information about requests to and responses from the Internet	About the Traffic View on page 38
Information about KVM and JAM status	MIDP
The source code of the content that is currently loaded in the SDK browser	About the Browser Source View on page 43
A history list of the SDK browser	About the Browser History View on page 45
Warning messages	About the Log View on page 46

About the Traffic View

The Traffic view displays events that occur between the SDK and an external entity such as a gateway simulator or the Internet. (To trace network events that occur in the KVM, see Setting MIDP Tracing and Speed Options on page 21.)

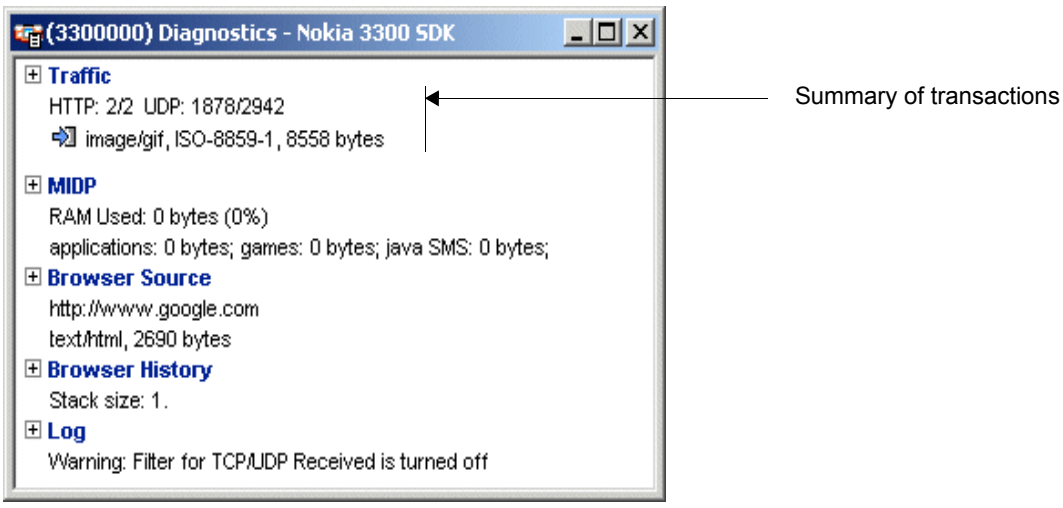
The events are triggered as the content on the SDK changes:

This event	Is triggered by
HTTP requests HTTP responses	• The browser sends or receives an HTTP request. • A MIDlet sends or receives an HTTP request using the HTTP Connection interface.
TCP stream data sending TCP stream data receiving	• A MIDlet sends or receives data through a `socket://` connection schema. • A MIDlet sends or receives an HTTP request through an `http://` connection schema when the SDK is configured to use the HTTP stack instead of WAP.
UDP datagram sending	• The browser sends an HTTP request. • A MIDlet sends an HTTP request through the HTTP connection when the SDK is configured to use the WAP stack.
UDP datagram reception	• The browser receives an HTTP response. • A MIDlet receives an HTTP response on an HTTP connection when the SDK is configured to use the WAP stack.
MMS message sending	An MMS message was sent using the phone menus.
MMS message reception	An MMS message was opened by the SDK or pushed by an external tool such as Nokia Developer's Suite for MMS 1.0 or Nokia Mobile Internet Toolkit 4.0 on the DSCLI interface.

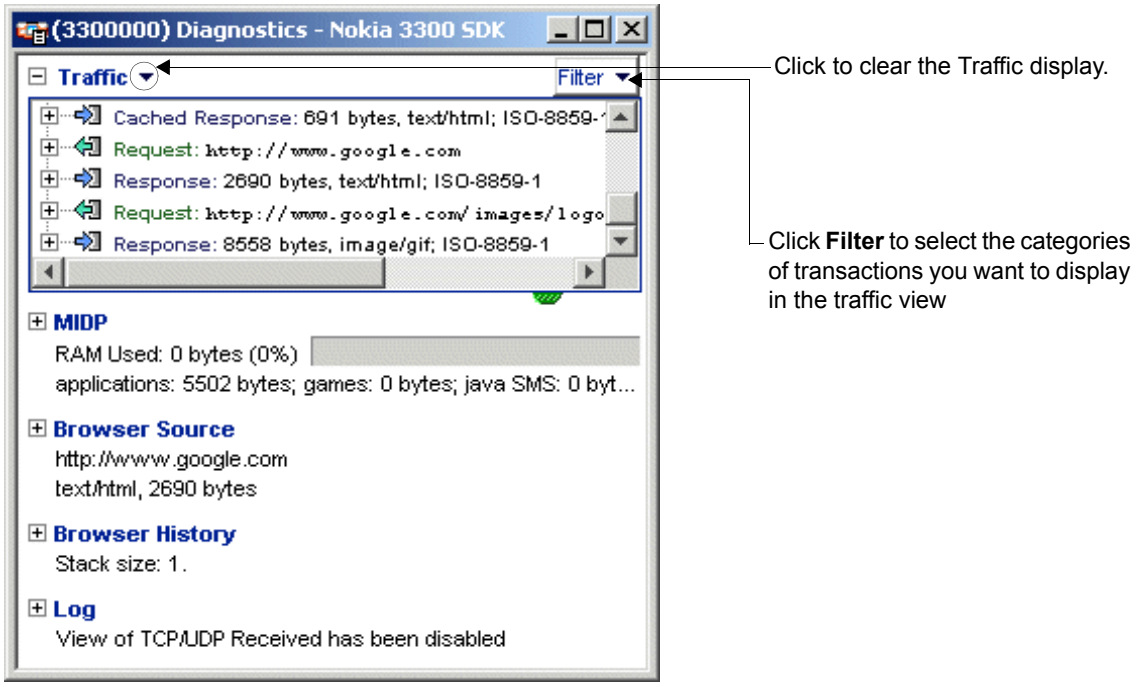
This event	Is triggered by
WAP push message reception	A WAP push message was opened by the SDK or pushed by an external tool such as Nokia Developer's Suite for MMS 1.0 or Nokia Mobile Internet Toolkit 4.0 on the DSCLI interface.
SMS message fragment sending SMS message fragment reception	• A short message fragment is sent to or through the **Messages>Create Message>Send** menu option or received from another client. • A MIDlet sends a short messages fragment through the `sms://` schema or receives a short message fragment on a short message connection.
SMS message sending SMS message reception	• A short message fragment is sent to another client through the **Messages>Create Message>Send** menu option or receives a short message from another client. • A MIDlet sends a short messages using the `sms://` schema or receives a short messages on a short message connection.

When the Traffic view is closed, this summary is displayed:

- Number of HTTP requests and responses (both WAP and HTTP)
- Number of bytes sent and received on the TCP layer
- Number of bytes sent and received on the UDP layer
- Number of SMS fragments (PDUs) sent and received



When you open the Traffic view, you see the list of transactions that have occurred and that occur while the browser is active:


(3300000) Diagnostics - Nokia 3300 SDK
Traffic
Filter
Cached Response: 691 bytes, text/html; ISO-8859-
Request: http://www.google.com
Response: 2690 bytes, text/html; ISO-8859-1
Request: http://www.google.com/images/logo
Response: 8558 bytes, image/gif; ISO-8859-1
MIDP
RAM Used: 0 bytes (0%)
applications: 5502 bytes; games: 0 bytes; java SMS: 0 byt...
Browser Source
http://www.google.com
text/html, 2690 bytes
Browser History
Stack size: 1.
Log
View of TCP/UDP Received has been disabled
Click to clear the Traffic display.
Click Filter to select the categories of transactions you want to display in the traffic view

The most recent transaction appears at the bottom of the list.

If an SMS message is shorter than 19 characters, the entire message is displayed. If it is longer than 10 characters, only the first 9 characters of the message is displayed. Binary SMS messages are not displayed.

To display only some of the transactions, click **Filter** and check only the categories of transactions you want to display. The information about the categories you don't check is available but undisplayed until you check them. The filter settings are persistent between instance invocations. The filter categories are:

Check this filter	For information about
Requests	Requests that originate in the SDK. For example, when the SDK sends a request to a web server, the transaction is recorded under Requests.
Content Responses	Responses made to the SDK as a result of a request the SDK made. For example, a web site responds to the SDK's request for it by loading on the browser, and the transaction is recorded under Content Responses.
Pushed Messages	Unsolicited transactions - that is, a transaction that is not a result of a request the SDK made. Examples of these transactions include Push and MMS messages that you can send to the SDK from IDEs, the CLI, or File>Open.

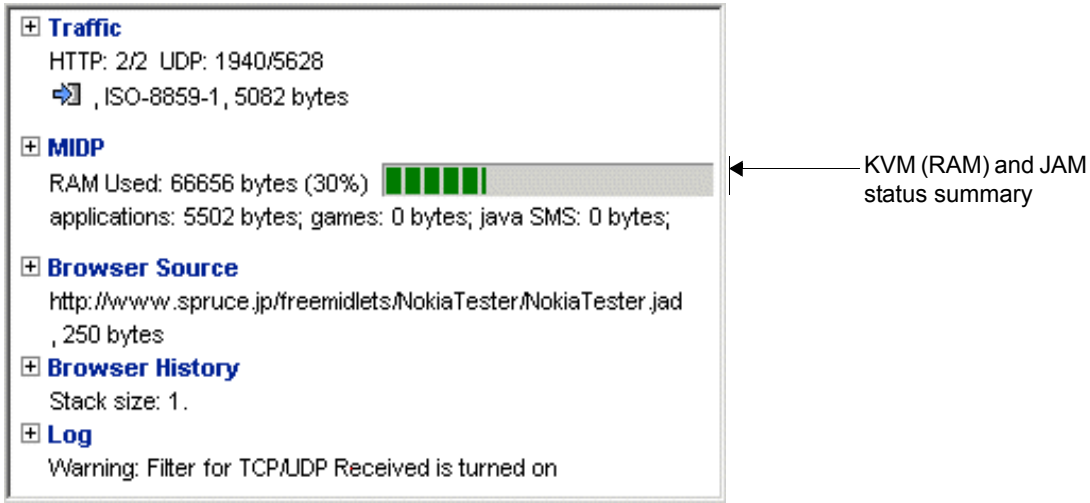
Check this filter	For information about
SMS Sent	Short Message Service Messages sent, including the size of whole messages and message fragments.
SMS Received	Short Message Service Messages received, including the size of whole messages and message fragments.
TCP/UDP Sent	User Datagram Protocol and User Datagram Protocol of messages the SDK sends. If you uncheck this filter, a note that this filter is inactive appears in the Log view.
TCP/UDP Received	User Datagram Protocol and User Datagram Protocol of messages the SDK receives. If you uncheck this filter, a note that this filter is inactive appears in the Log view.

About MIDP View

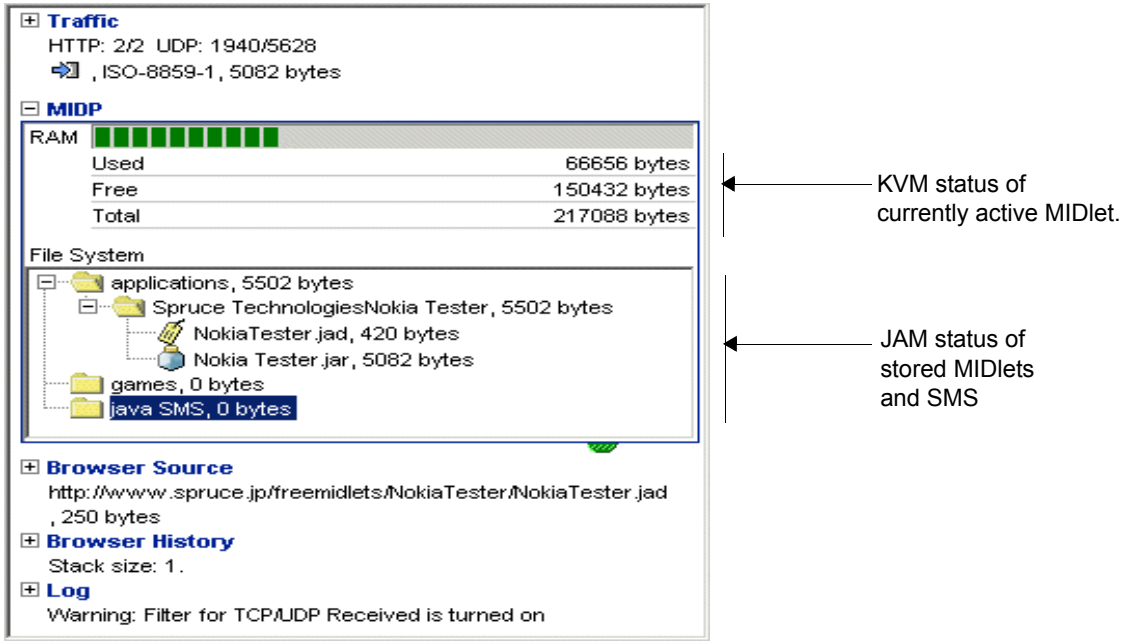
The MIDP view provides information about:

- JAM (Java Application Manager) storage size; that is, how much space provisioned MIDlets occupy; the size of unprovisioned MIDlets is not recorded. JAM storage size does not take into account how much additional space MIDlets require to run.
- KVM (kilobyte virtual machine) is software that is an interface between Java binary code and the microprocessor on a computer that actually performs the program's instructions. The size of the KVM memory heap (heapsize) controls how much memory a MIDlet can take to run on the SDK. (See About KVM Heapsize Status on page 42.) The KVM status is updated as a MIDlet runs.

When the MIDP view is closed, the Diagnostics window looks like this:



When you open the MIDP view, you see the current status of JAM and KVM storage:



About KVM Heapsize Status

For the MIDlet that is currently active on the SDK, the KVM memory status shows in bytes:

- **Free**: Bytes in the KVM heap that are available for use.
- **Used**: Bytes of the KVM heap that are in current use.
- **Total**: The total size of the KVM heapsize. (To modify the size of the KVM heapsize, see Setting MIDlet KVM Speed on page 22.)

About JAM Storage Status

Click a folder. If a folder has content, it opens to display:

- The MIDlets that are stored in the folder
- The name and total size in bytes of the MIDlet suite, which consists of JAR and JAD files
- The size of the JAD file in bytes
- The size of the JAR file in bytes
- The status of SMS messages that are generated by MIDlets

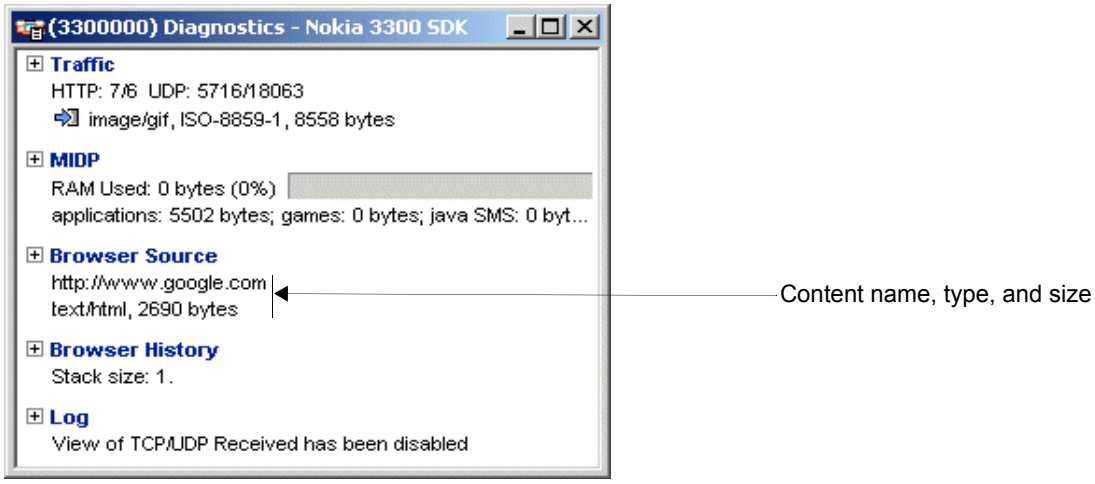
A folder that is empty remains closed when you click it.

The storage size of a file reflects the size of the file on the handset without the file system overhead of file headers. This size, as it is reported in the Diagnostics window is always larger than the actual data size of the file.

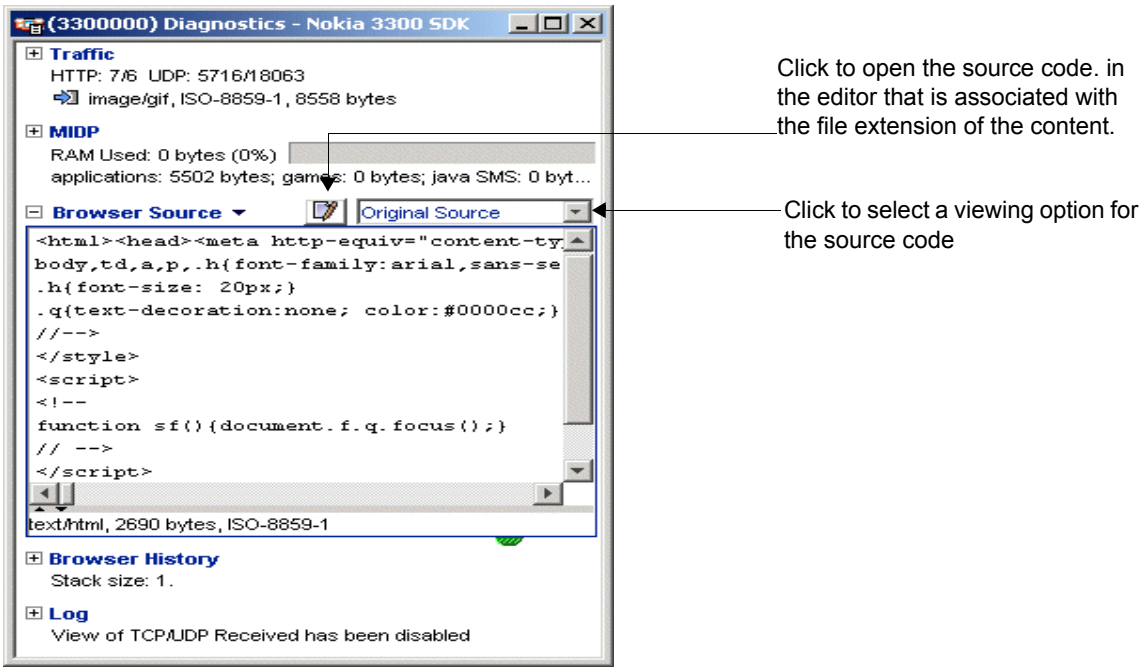
About the Browser Source View

The Browser Source view displays the source code of the content that is currently loaded on the SDK browser.

When the view is closed, you see the name of the current content, its file type and size.



When you open the Browser Source view, you see the source code of the content that is on the browser:



Click the **Edit** icon to open the source code in the editor or browser that is associated with the file extension of the content. (To change the default file association, go to **Folder Options>File Types** in the Windows Control Panel.)

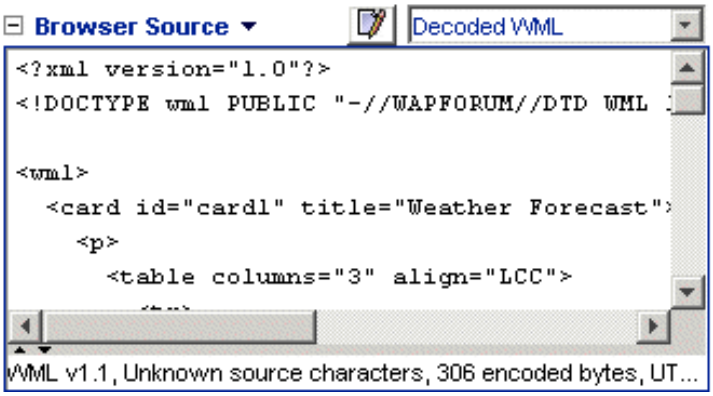
You can view the current content in several formats, depending on the content that's on the browser. To see what views are available, click on the drop-down menu on the top-right corner of the Browser Source view. These options are available under these circumstances:

If the current content is	You can view the source as
Unencoded, as in XHTML or WML	Original source code
Encoded, as in binary WML (`.wmlc`)	Decoded WML
	Element Tree
	Byte Code

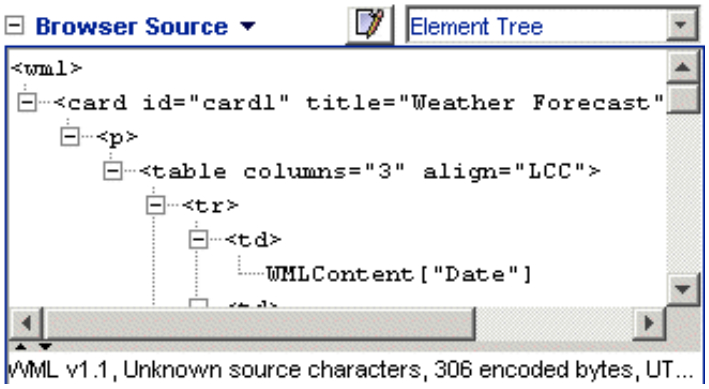
Nokia Mobile Internet Toolkit (NMIT) always encodes WML content into WMLC before passing it to the SDK. Therefore, the Original Source view is not available when you are using source that comes from NMIT.

When content is encoded WML, you can choose from among three views:

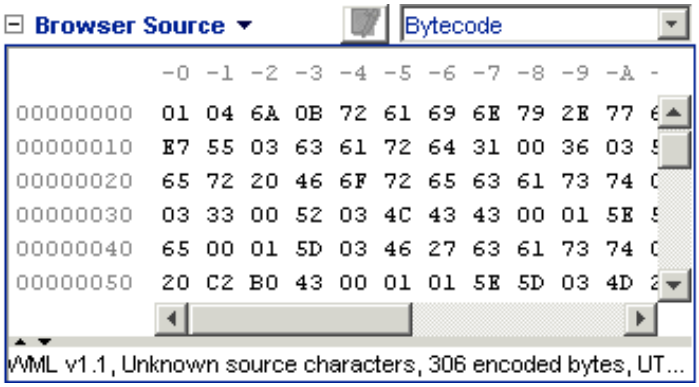
- The **Decoded WML** view displays the result of decoding an encoded response received from a server.



- The **Element Tree** view displays a parse tree of the current content as the SDK received it. This view is useful for detecting the hierarchical structure of the content.



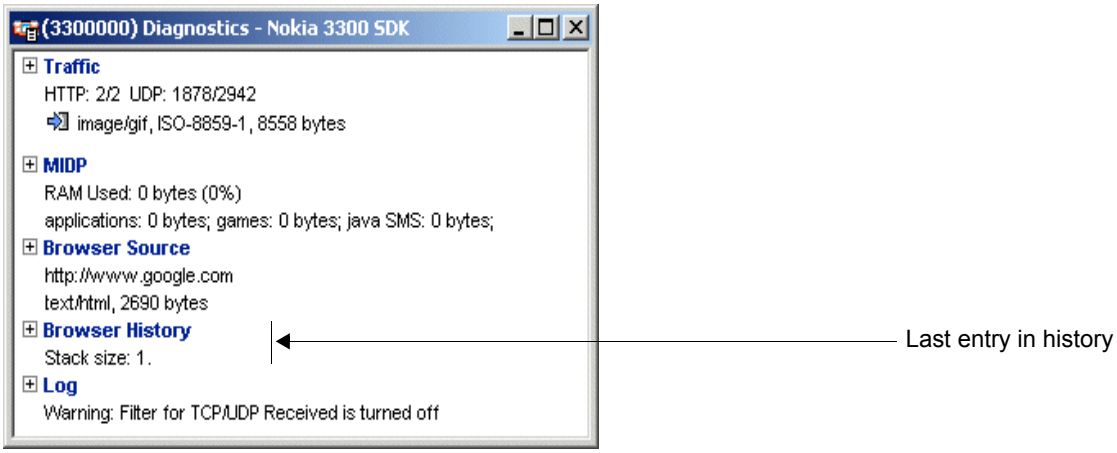
- The **Bytecode** view displays the bytes the SDK receives. Some content types such as images are available only in bytecode. You cannot edit Bytecode.



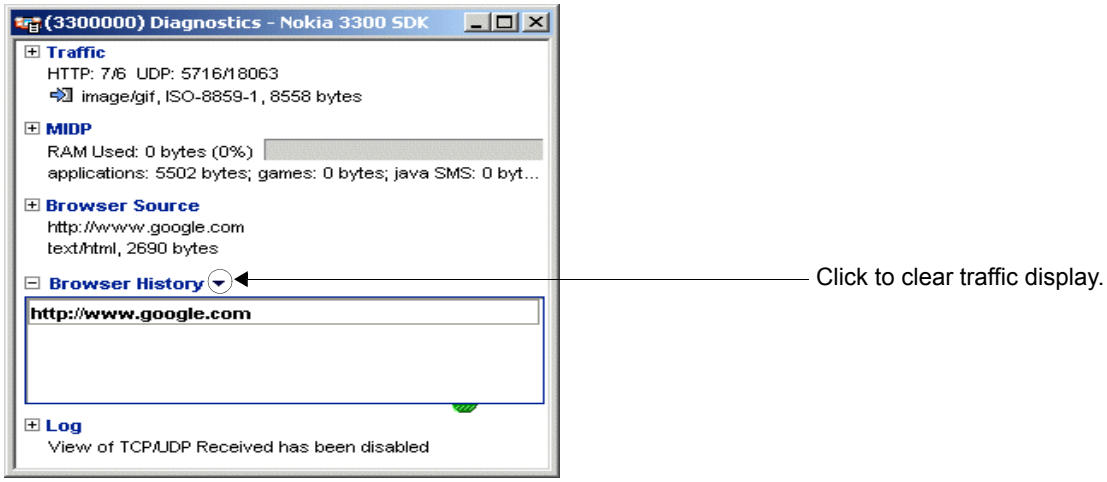
About the Browser History View

The Browser History view displays the list of sites that have been displayed on the Browser during the current session.

When the Browser History view is closed, you see a summary of the number of sites listed in the open view:



When you open the view, you see the list of locations with the most recent location at the bottom. The location that is currently displayed on the browser is highlighted:



The Browser history on the SDK is erased when you:

- Quit the browser.
- Open local content using **File>Open.**
- Browse to new content using **File>Open.**
- Preview new content from an IDE.

To preserve the browser history when you change web sites, change web sites with the menus on the SDK (as you would on the handset). For example:

1 Select **Menu>Services>Bookmarks>Options>New Bookmark** and enter: `http:\\wap.yahoo.com`. Then select **OK** and enter `Yahoo` for the title. Select **OK.**

2 Select **Options>New Bookmark** and enter: `http:\\wap.ayg.com`. Then select **OK** and enter `AYG` for the title. Select **OK.**

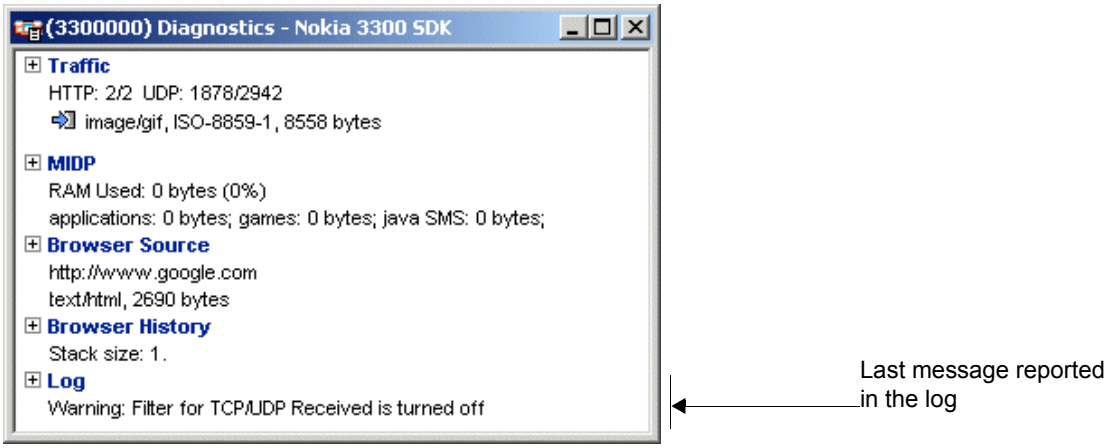
3 Select **Yahoo>Options>Go To. Yahoo.com loads on the browser, and the browser history records the URL.**

4 Select **Options>Bookmarks>Google>Options>Go To. AYG.com loads on the browser, and the browser history records the URL without deleting the Yahoo listing.**

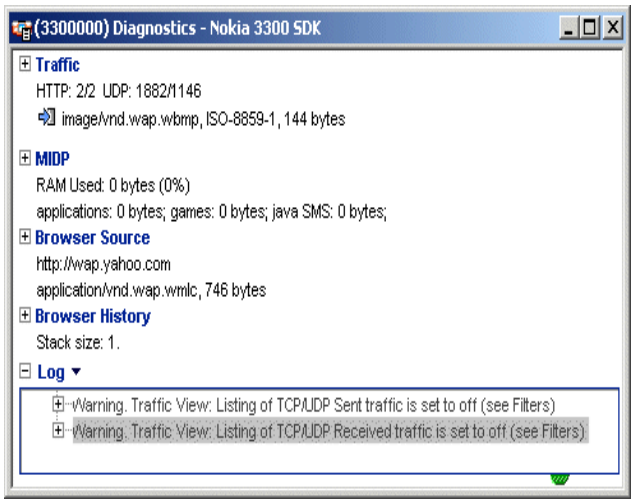
About the Log View

The Log view displays messages that the SDK generates, such as warnings about unsupported content types and file sizes that exceed what the SDK can handle. The Log will also tell you whether the TCP/UDP filters in the Traffic view are turned off.

When the Log view is closed, it looks like this:



When you open the Log view, you see any messages that the SDK generated:


(3300000) Diagnostics - Nokia 3300 SDK
Traffic
HTTP: 2/2 UDP: 1882/1146
image/vnd.wap.wbmp, ISO-8859-1, 144 bytes
MIDP
RAM Used: 0 bytes (0%)
applications: 0 bytes; games: 0 bytes; java SMS: 0 bytes;
Browser Source
http://wap.yahoo.com
application/vnd.wap.wmlc, 746 bytes
Browser History
Stack size: 1.
Log
Warning. Traffic View: Listing of TCP/UDP Sent traffic is set to off (see Filters)
Warning. Traffic View: Listing of TCP/UDP Received traffic is set to off (see Filters)

What's Next?

Go ahead and begin:

- Developing MMS and Push messages, as described in Using the SDK for Message Content Development on page 49
- Developing MIDlets, as described in Using the SDK for MIDlet Development on page 59

Or try running the SDK from a command line. See Running the SDK from a Command Line on page 67.

5

Using the SDK for Message Content Development

You can create Multimedia Messaging Services (MMS), Service Indicator (SI), and Service Load (SL) messages on an application and send them to the Nokia 3300 SDK within an integrated development environment.

When you create an MMS message by using the SDK's phone menus (as a user of a Nokia 3300 handset would do), the message is not saved to a file because any message you create on the SDK is stored in the SDK's memory and is deleted when you terminate the SDK instance. To save an MMS message, you must either:

- Create and save the message in another application before displaying the message on the SDK.
- Use the `-Xoutbox` option from the command line interface. This option makes the SDK send and save the message into a directory you specify. See Running the SDK from a Command Line on page 67.

About MMS Messages

An MMS message contains rich media content that may include one or more text, image, or audio component. An MMS message can optionally also include a SMIL (Synchronized Multimedia Integration Language) component that specifies the layout, sequence, and timing of the parts in the MMS message on a handset.

Nokia 3300 SDK can handle an MMS message whose content contains one or more parts of these content types:

Content Type	File format	Can Only Be Received?
`image/jpeg`	JPEG	
`image/gif`	GIF87a, GIF89a	
`image/png`	PNG	
`image/bmp`	BMP	
`image/vnd.wap.wbmp`	WBMP	
`image/vnd.nok-wallpaper`	wallpaper	Yes
`audio/midi`	MIDI	Yes
`audio/mid`		
`audio/x-mid`		
`audio/sp-midi`	MIDI	Yes
`audio/x-midi`	MIDI	Yes
`audio/amr-wb`	MIDI	
`text/plain`	text	

Some content types can only be received; they cannot be forwarded. For example, you can't forward a MIDI content type (`audio/midi`) that contains a ring tone you've purchased.

Creating an MMS Message Directly on the SDK

You can simulate how a user creates an MMS message on a handset by clicking the keys on the SDK. To create an MMS message directly on the SDK:

1 Select **Menu**, then select **Messages**.

2 Select **Multimedia Msgs>Create Message** and type the text of the message.

3 Do one or both of the following:

- Select **Options >Insert Image>Select>Graphics** folder, and then select the graphic you want.
- Select **Options >Insert Image>Select>Tones** folder, and then select the sound you want.

4 Select **Open**.

5 Select one of the following:

- **Send to Number** and enter a telephone number.
- **Send to Email** and enter an email address.

6 Select **OK** to send the message to the outbox folder.

Creating and Previewing an MMS Message Using NMIT 4.0

Creating a new MMS message within NMIT is a two-stage process:

1 In NMIT, create the MMS message with the MMS Set Up Wizard.

Select **File>New>Messaging>MMS Message** to open the MMS Setup Wizard. Within the wizard, specify the recipients of the message, the media files to be included in the message, and whether to include or create a SMIL file. The wizard supplies required headers with default header values, which you can change in Step 2. Click **Finish** in the Wizard.

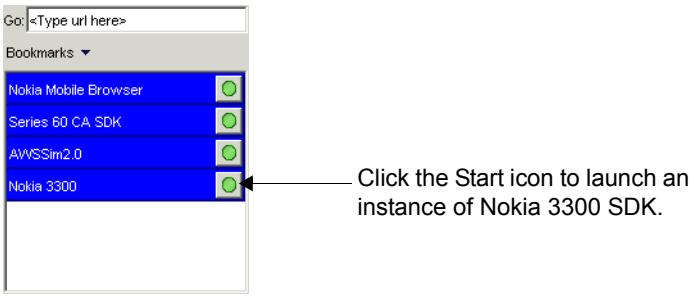
2 In NMIT, edit the MMS message header.

When you click **Finish** in the wizard, NMIT opens the MMS Message Multipart editor where you can edit the values of the headers, both for the entire MMS message and for the individual parts. You can also add optional headers.

To preview an MMS message on the SDK with NMIT 4.0:

1 Select **Tools>SDK Control Panel** to display phone SDKs that NMIT has detected.

2 To start a phone SDK, click the Start icon next to its name.



3 NMIT starts the Nokia 3300 SDK and the panel displays the SDK instance identifier. (To start another instance of the Nokia 3300 SDK, click the Start icon again.)

4 Click **Push** within the MMS Message Editor to send the MMS message to the SDK's Message Inbox.

The SDK displays a note that an MMS message has arrived. If the SDK is in browsing mode, the MMS message announcement is not displayed, and you must use the phone menus to quit the browser to see the announcement.

To close an SDK instance, select **File>Exit** from the SDK, even if you have started the instance from within NMIT.

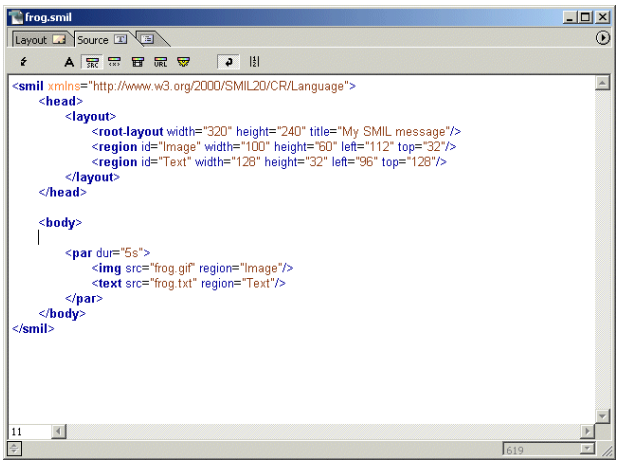
Creating and Sending an MMS Message With Adobe GoLive and NDS for MMS

To create an MMS message in Adobe GoLive, you edit a SMIL document by entering the file paths for all media parts to be included in the MMS message. Then, you use NDS for MMS under the **Nokia Tools** menu to wrap the MMS message with MMS headers and you save the message.

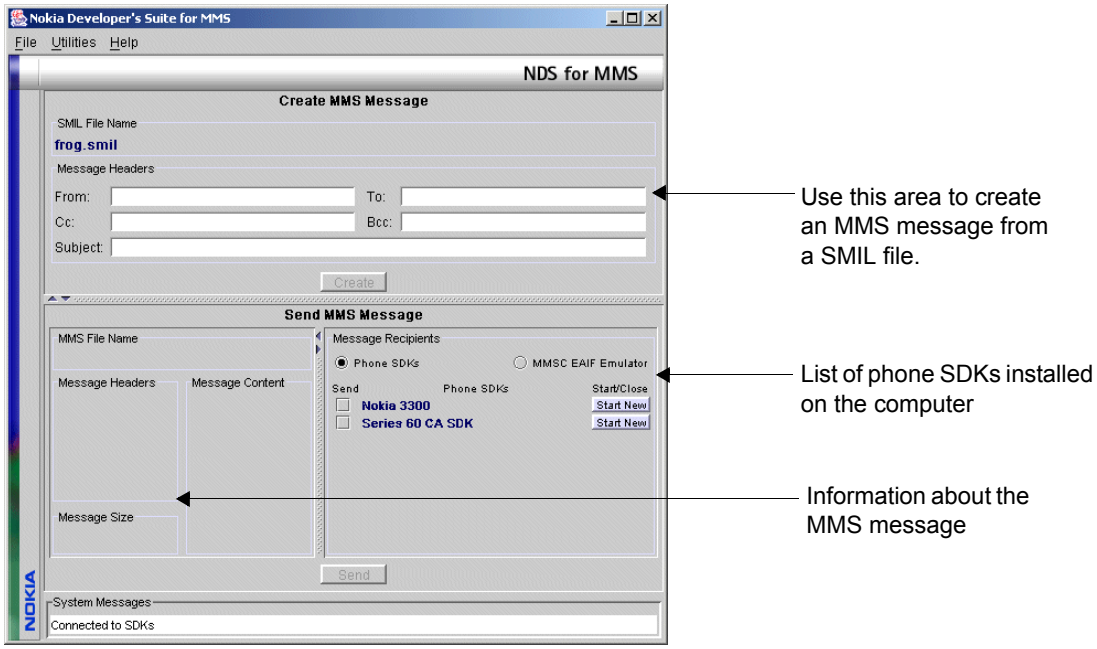
To view MMS messages on the SDK using NDS for MMS within Adobe GoLive:

1 In Adobe GoLive, do one of these steps:

- Select **File>New Special>SMIL Document** to create a SMIL file. Adobe GoLive opens a SMIL document template in an editor window.
- Select **File>Open** to open a `.SMIL` file like the one below:



2 In Adobe GoLive, select **Nokia Tools>Create\Send MMS Messages.** NDS for MMS starts:



3 In NDS for MMS, fill in the **To** and **From** fields and click **Create**.

4 In NDS for MMS, select the SDK to which you will deploy the message.

5 In NDS for MMS, click **Send**.

The SDK displays a note that a message has arrived. If the SDK's browser is active, the MMS message announcement is not displayed, and you must use the phone menus to exit from the browser to see the announcement. You know you have exited from the browser when you see Menu and Names on the display.

Processing an MMS Message on the SDK

This example shows you how to receive, view, and dispose of an MMS message in the SDK. The MMS message in this example consists of two media parts, a TXT file and a GIF image. The message also contains a SMIL file that coordinates the timing of the TXT and GIF files; however, like the Nokia 3300 handset, the Nokia 3300 SDK does not use all the information in the SMIL file. The SDK displays the TXT and GIF files in the order they appear in the MMS file.

1 Make sure you have exited from the browser so you can see the announcement that is displayed when the SDK receives the MMS message. You know you are out of the browser when you see **Menu** and **Names** on the display.

2 When the message arrives, the SDK displays an announcement.

3 Click the left soft key to select **Show** and to display the first part of the MMS message, `frog.txt`.

4 Click the down arrow key to display any other components of the MMS message. The first component (`frog.txt`) is redisplayed when you reach the end of the components.

In this example, `frog.gif` is displayed after `frog.txt`.

5 Click the left soft key to select **Options** and display the options for managing the message:

- Reply - reply to the sender of the message.
- Forward to no. - forward message to the phone number you specify.
- Forward to email - forward message to an email you specify. (The SDK does not permit a genuine forward.)
- Message Details - display information about the message.
- Play - play the message.
- Save image - save the image in the message.
- Delete message - delete the MMS message.

Working with Push Messages

An SI (service indicator) message is a text message sent to a phone to notify the user that a service or resource is available. The message contains a URL that identifies the resource. The SDK displays this text notification and the user can choose to access the resource or to delete the notification.

An SL (service load) message automatically loads a URL on the user's phone.

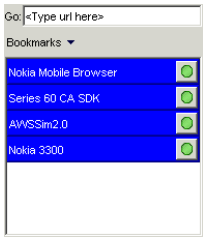
Because the main content of a Push message is the URL, the handset offers no **Save** option for Push messages. SI Push messages stay in the Service Inbox until you delete them. You cannot save the same Push message again until you delete the first one.

Creating a Push Message with NMIT

You can use Nokia Mobile Internet Toolkit 4.0 (NMIT 4.0)to preview SI or SL push message on the SDK. The SDK used as standalone application or with Adobe GoLive does not support Push messages.

This example shows how to create and push an SI message from NMIT 4.0:

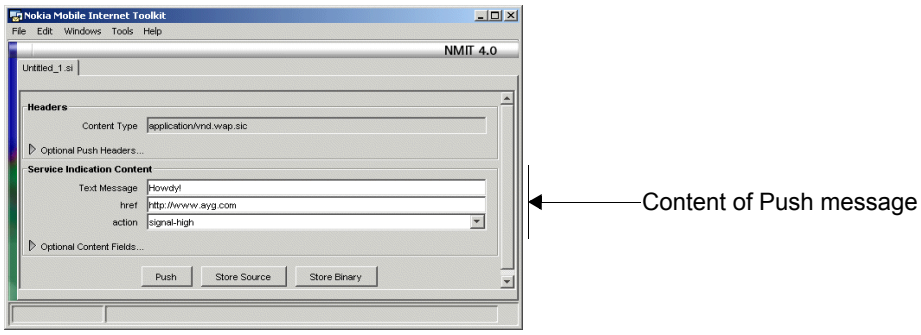
1 In NMIT 4.0, select **Tools>SDK Control Panel** and start and instance of the Nokia 3300 SDK as the active device.



2 In NMIT, do one of the following:

- Select **File>Open** from NMIT's main menu or from the Toolkit Editors window and select a local SI or SL file. Click **Open**. NMIT displays the SI file in the Toolkit Editors window.
- Select **File>New** from NMIT's main menu or from the Toolkit Editors window. Enter the text you want for the text announcement in the **Text Message** box and the URL to be retrieved in the **href** box.

  With either choice, this dialog box appears:



3 Click **Store Binary** to create an SIC Push message.

4 Click **Push** to send the message to the SDK.

Processing a Push Message on the SDK

This example shows how to process on the SDK the SIC Push message you created in the previous section:

1 Make sure you have exited from the browser so you can see the message that is displayed when the SDK receives the Push message. You know you are out of the browser when you see **Menu** and **Names** on the display.

2 When the SDK receives the message, the SDK displays an announcement. Select **Show** to display the text part of the message,

3 The text of the message appears (*Howdy!*). Select **Options** to display ways of processing the message.

4 Select one of the following:

- **Retrieve** to load the URL in the href into the browser. In this example, the URL is *www.ayo.com*, as shown on the right.
- **Details** to display information about the message.
- **Delete** to delete the message.

5 If you select **Retrieve**, the SDK connects with *www.ayg.com*, the URL defined within NMIT under **href**.

Some Push messages do not create an announcement because their priority level is not high enough. To view the content of such a message, you'll have to navigate to the Service Inbox and select the message.

What's Next?

You can get information about the content on the Nokia 3300 SDK. See Getting Information About the Content on the SDK on page 35.

Or you might want to explore running the SDK from a command line. See Running the SDK from a Command Line on page 67.

6

Using the SDK for MIDlet Development

The Nokia 3300 SDK replicates the appearance and function of MIDlets as they are run on the handset by using the same Java virtual machine and support classes that the Nokia 3300 handset uses.

The Nokia 3300 SDK uses a J2ME environment that supports CLDC 1.0, MIDP 1.0, and the Nokia UI API.

About MIDP MIDlets

A MIDlet is a Java application that conforms to the Mobile Information Device Profile (MIDP) specification. MIDP applications are portable across cell phones and can handle differences in screen size and button layout, as well as the low memory and small footprint of a cell phone

Typically, a MIDlet is a game or application that can be downloaded from the Internet and used on a mobile phone. A MIDlet can be made up of:

- JAD (Java Application Descriptor) file, which describes the attributes of the JAR file. One function of the JAD file is to let an SKD or device determine whether it has the capability to run the JAR file. A JAD file can contain information such as:

```
MIDlet-Name: Bounce
MIDlet-Vendor: Nokia
MIDlet-Version: 1.0.1
MIDlet-Jar-Size: 25978
MIDlet-Jar-URL: http://domain/directory/Bounce.jar
Nokia-MIDlet-Category: Game
MIDlet-1: MIDletName, MIDletIcon, MIDletMainClassName
```

- JAR (Java ARchive) file, which contains the actual preverified MIDlet. A JAR file contains the class, image, and sound files gathered into a single file and compressed for faster downloading to your SDK or handset.
- A JAR manifest file (`manifest.mf`)that contains information similar to the information in the JAD file. The JAR manifest file is located within the JAR file and is available only after you've downloaded the JAR file.

For more information about MIDP MIDlets, see:

- *Mobile Information Device Profile (JSR-32) 1.0*, freely available from www.forum.nokia.
- *Connected, Limited Device Configuration (JSR 30) 1.0.4*, freely available from www.java.sun.com

About Loading MIDlets With and Without Provisioning

You can load a MIDlet on the Nokia 3300 SDK:

- Without provisioning - that is, by opening a MIDlet from a local hard drive to the SDK.
- With provisioning - that is, by downloading a MIDlet from the Internet with the SDK before running it on the SDK with the browser.

Running a MIDlet without provisioning provides a faster, less restrictive environment in which to test MIDlets during early development. Provisioning a MIDlet can be more constraining but truer to what a user does when using a Nokia 3300 handset. Provisioning also provides a validation process that ensures the MIDlet is usable on the SDK. Here is a comparison between provisioning and not provisioning a MIDlet:

MIDlet trait	With provisioning	Without provisioning
Validated JAD and JAR files	Yes	No
MIDlet installed in the SDK instance's Games or Applications folder	Yes	No
Maximum size of JAD file	5000 bytes	300 kb

MIDlet trait	With provisioning	Without provisioning
Maximum size of JAR file	• 32,200 bytes if over-the-air downloaded • 63,720 bytes otherwise	300 kb (Can be changed in **Tools>Preferences>MIDP**

Running a MIDlet Without Provisioning

Running a MIDlet without provisioning on the SDK:

- Lets you develop a MIDlet from within an IDE during which you edit and compile code and then immediately display the results on the SDK.
- Is a good way to test how a user runs a MIDlet that is already loaded on a Nokia 3300 handset.

You can run a MIDlet without provisioning by:

- Using the command line interface without the `provisioning` option to open a MIDlet from a local hard drive. For example: `emulator.exe bounce.jar`. (See Running the SDK from a Command Line on page 67.)
- Using **File>Open** and entering the pathname of a local JAD or JAR file.
- Sending a MIDlet directly to the SDK from an application within an IDE.

Provisioning a MIDlet

Any MIDlet that is downloaded from an HTTP site using the SDK's browser is always provisioned.

Provisioning a MIDlet lets you test how an enduser experiences downloading a MIDlet from an Internet site and provides MIDlet validation - a series of integrity checks that ensure the SDK can run a MIDlet. MIDlet validations involves checking the consistency of various attributes in the JAD and JAR manifest files of a MIDlet.

Using the SDK, you can provision a MIDlet in these ways:

- You can emulate provisioning using files on your local drive by starting the SDK from the command line with the `--Xprovisioning` option. For example:
  `emulator.exe -Xprovisioning mylocalmidlet.jad`

  (See Running the SDK from a Command Line on page 67.)
- You can use **File>Open** to enter the URL of a Web site that hosts a JAD or JAR file.
- You can post your MIDlet on an Internet server or server simulator and use the SDK's browser to navigate to it by clicking the keys on the SDK.
- You can select the provisioning mode by selecting **Tools>Settings** and clicking the **MIDP** tab to select the provisioning option. For more information, see Setting MIDP Provisioning and Execution Options on page 19.

Provisioning Using the Network

To provision a MIDlet using the network, make sure a network connection is available.

To configure an HTTP proxy, do one of the following:

- Start the SDK on the command line using the `-Xhttp_proxy` option. For example:
  `emulator.exe -Xprovisioning -Xhttp_proxy 113.14.176.11:8080 mymidlet.jad`

  (See Running the SDK from a Command Line on page 67.)
- Start the SDK and select **Tools>Settings>HTTP Networking**. Specify the address and port on the HTTP networking tab. See Setting Up Network Access on page 17.

After configuring the network connection, you can provision the MIDlet. For example, Select **File>Open** and enter the URL of a site hosting a MIDlet - for example, http://www.spruce.jp/freemidlets/NokiaTester/NokiaTester.jad.

Based on the information in the JAD file, the Java Application Manager (JAM) on the SDK determines whether the SDK can run the MIDlet. If it can, you can elect to load the JAR file of the MIDlet, which contains the actual application, and save it under **Applications** or **Games**.

The MIDlet is installed in the **Games** folder only if the JAD file contains the string `Nokia-MIDlet-Category: Games`. Otherwise, by default, the MIDlet is installed in the Applications folder. The JAM might optionally forward a notification to the Internet server that the MIDlet has been downloaded (usually done for the purposes of billing or customer care).

Provisioning Using Local Files and the Command Line Interface

True MIDlet provisioning requires an Internet connection to a URL of a web site that hosts a JAD file. However, you can simulate the provisioning of a MIDlet by starting the SDK from the command line with the `-Xprovisioning` option and with a MIDlet file residing on a local drive, as described here. (For more information about the command line interface, see Running the SDK from a Command Line on page 67.

To provision a MIDlet using a local file and the CLI:

1 From the command line, enter a command like this one:
`emulator.exe -Xprovisioning mylocalmidlet.jad`

2 The SDK starts and displays the information in the JAD file about the MIDlet.

3 Based on the information in the JAD file, the Java Application Manager (JAM) on the SDK determines whether the SDK can run the MIDlet. If it can, you can elect to load the JAR file of the MIDlet, which contains the actual application. The SDK provides you with an opportunity to save it.

About JAD and JAR Validations for Provisioned MIDlets

When MIDlets are provisioned, the SDK checks (validates) the JAD, JAR, and JAR manifest files. When you provision a JAD file, the JAR and JAR manifest files associated with it are validated. When you provision a JAR file, only the JAR manifest file is validated.

The checks occur in a specific order, and the SDK stops performing the checks when the first check fails, after which the SDK displays a warning and discards the JAD or JAR file.

The SDK checks that:

- The `JARSize` attribute is not greater than 63,720 bytes.
- The JAD and JAR size are consistent.
- The value in the `MIDlet-Data-Size` attribute is not greater than the value specified in the JAD file.
- The attributes defined in `MIDlet-Name` and `MIDlet-Vendor` are the same in the JAD file and the JAR manifest file.
- The `MIDlet-Version` attributes defined in the JAD file and in the JAR manifest file match.
- The `MicroEdition-Configuration` attribute defined in the JAR manifest file is `CLDC-1.0`.
- The `MicroEdition-Profile` attribute defined in the JAR manifest file is `MIDP 1.0`.

Setting Up the SDK To Run MIDlets

Before you run a MIDlet on the SDK, you can set up persistent setting that:

- Control whether MIDlet s are provisioned. See Setting MIDP Provisioning and Execution Options on page 19.
- Provide event tracing and MIDlet speed control. See Setting MIDP Tracing and Speed Options on page 21.
- Control how MIDlets access the network. See Setting Up Network Access on page 17.
- Let you set the KVM heapsize that controls how large a MIDlet you can store and run. See Setting the KVM Heapsize on page 20.

Using the Nokia 3300 SDK With JBuilder

After you have installed and configured JBuilder, you can begin to use the SDK within its environment. For more information, see the *Nokia 3300 SDK Installation and Configuration Guide*.

About the Nokia 3300 SDK Life Cycle in JBuilder

In JBuilder, an SDK instance is associated with a JBuilder project. The SDK configuration parameters associated with an instance are stored on a per project basis in the JBuilder project file (`*.jpx, *.jpr`). When you choose **Run Project** or **Debug Project**, a new SDK instance starts and runs the MIDlet using the configuration parameters already stored for that project.

Creating a MIDlet With JBuilder and Running It on the SDK

To create a MIDlet from scratch and then run it on the SDK, do this from within JBuilder:

1 Choose **File>New Project** to create a new project. Complete JBuilder's **Project Wizard**.

2 Choose **File>New** to create a new MIDlet. Select the **Micro** tab and choose the MIDlet template (**MIDlet**).

3 Choose **Run Project**. A new SDK instance starts and runs the MIDlet you've created. No configuration parameters are requested.

4 Click **Reset** to close the SDK instance.

Running a MIDlet and Displaying Trace Information in JBuilder

To enable tracing for the active project without affecting other JBuilder projects, do this from within JBuilder:

1 Choose **Project>Project Properties**.

2 In the **Project Properties** dialog box, click the **Run** tab.

3 Click **Edit** to the runtime configuration in the **Runtime Configuration Properties** dialog box.

4 Click the **MIDlet** tab and select the MIDlet you want from the drop-down menu on the **MIDlet** tab.

5 In the **Emulator parameters** field, enter one of the command line tracing options such as `-Xverbose:method`. Click OK to close this dialog box. (In fact, you can enter any of the CLI options in this box except `-new`, `-use:`, `-shell`, and `-qinst`. See Options for General Use on page 71.)

6 Select **Run Run>Project**. The tracing information is shown in JBuilder's **Output** window.

For more information about command line interface options, see Running the SDK from a Command Line on page 67.

Using the SDK with JBuilder and NDS for J2ME

To run the SDK with JBuilder and NDS for J2ME, follow these general steps:

1 In JBuilder, create a new project or open an existing project, and then compile the Java files.

2 In JBuilder, choose **Tools>Nokia Developer's Suite for J2ME>Create Application Package**. The NDS Create Application Package dialog box is displayed. Enter the necessary information (see the NDS for J2ME User's Guide) and click **Finish**. You now have a JAD and a JAR file.

3 Select the JAD file and choose **Tools>Nokia Developer's Suite for J2ME>Start Emulators**. The MIDlet is run on the SDK you've selected.

Using the Nokia 3300 SDK With Sun One Studio

After you set the Nokia 3300 SDK as the default emulator, you can proceed with any operation on Sun One Studio and the operation will use the default device you've chosen. For more information, see the *Nokia 3300 SDK Installation and Configuration Guide*.

What's Next?

You can get information about the content on the Nokia 3300 SDK. See Getting Information About the Content on the SDK on page 35.

Or you might want to explore running the SDK from a command line. See Running the SDK from a Command Line on page 67.

7

Running the SDK from a Command Line

Although anyone can use the SDK through the Windows command line, you'll be more likely to use the command line interface if you are developing browser or message content. If you are developing MIDlets, you'll likely use command line options within a Java IDE (for example, to trace MIDlet events).

Configuration options you set on the command line override those you set in the **Tools>Settings** dialog box but are not persistent between instances of the SDK. If you need settings to be persistent, use the settings in the **Tools>Settings** dialog box.

Accessing the Command Line Prompt

To access the command line prompt:

1 Select **Start menu>Run**.

2 Enter `explorer` in the **Run** dialog box.

3 In the Explorer window that appears, navigate to the directory where you installed the SDK.

4 Right-click on the `\bin` directory, and select **Command Line Prompt Here** from the drop-down menu.

In the command line window that appears, enter the command for starting the SDK, which is covered in the next section.

About the Command Line Syntax

The syntax for the command that starts the SDK is:

```
[<installation directory>\bin\<executable>] [...-option] [parameter]
```

This table summarizes the components of the syntax:

Component	Description
`executable`	The SDK can supports two executables: • `emulator.exe` – Runs UEI-compliant options,which are generally used for MIDP MIDlet development. • 3300.exe – Runs mostly non-UEI-compliant options and parameters that are usually used in message development. See About the Executables on page 69.
`-option`	Options generally specify configuration settings for the SDK such as the language it uses in each SDK instance and what events to trace. Most options work with `emulator.exe` and `3300.exe`; however, some options work only with one of the executables. Those options are noted. See About Options on page 69.
`parameter`	Parameters either load content or close an instance. Parameters work only with the `3300.exe` executable. You can have only one parameter per command line. Separate a parameter from any other element with spaces. The parameter and any options that go with it must be the last item in the command. The two parameters you can use are `load` and `close`. See About Parameters on page 74.

About the Executables

The SDK supports two executables:

- `emulator.exe` - Runs UEI-compliant options,which are generally used for MIDlet development.
- `3300.exe` - Runs options and parameters that are not UEI-compliant and are usually used in browser content development.

About emulator.exe

Typically, an IDE that supports MIDlet development automatically uses `emulator.exe` when it uses the SDK as a device. A developer of MIDlets is unlikely to use `emulator.exe` on the command line to monitor the behavior of a MIDlet. Options that you can use only with `emulator.exe` are noted in the option listing under About Options on page 69. Options and parameters that emulatore.exe does not support are ignored. IDEs can automatically detect which executable to use.

About 3300.exe

The executable `3300.exe` supports options and parameters that you use in message content development. If you are developing message or browser content, you are more likely to use 3300.exe on the command line because with this executable you can work within a shell (see Working With a CLI Shell to Repeatedly Send Commands on page 75) and arrange to have one instance of the SDK send a message to another instance (see Using Inboxes and Outboxes With MMS Messages on page 75).

Options that you can use only with `3300.exe` are noted in the option listing under About Options on page 69.

Options and parameters that 3300.exe does not support cause the entire command to fail, and an error message to be posted.

About Options

The SDK interprets all text preceded by a dash (-) as an option. You can specify multiple options, each one preceded by a dash, with a space between each option, as well as between an option and any subsequent or preceding option value. If a value contains one or more space characters, the value must be enclosed between quotation marks (“).

For example, the following command starts the SDK, specifying it should always remain on top of other windows and that it should display menu names in Finnish:

```
emulator.exe -Xontop -Xlanguage:Finnish
```

Some options work only with `emulator.exe` or `3300.exe`. These options are specifically marked with the name of the executable that supports them. Options work with both executables are simply listed. Options that `emulator.exe` does not support are ignored when you use them, and the command continues to execute those options it supports.

Options that `3300.exe` does not support cause an error message to be displayed and the command to fail.

Getting Information About the CLI Options

To display a summary on the screen of the command line options in the Nokia 3300 SDK, type: `emulator.exe -help or 3300.exe -help`. Depending on the executable you use, the command displays in the DOS Console a list of the options that work with the executable. The `-help` option must be the only option in a command.

Options for Managing SDK Instances

You can use these options to manage SDK instances.

Option	Description	Use for
`-Xlanguage:<language>`	Specifies the language that the SDK should use for its menu names.	Browser MMS MIDP
`-new` `(Use with 3300.exe only)`	Creates a new SDK instance. Do not use if you are using `-use` in the same command.	Browser MMS MIDP
`-use:<instance ID>` (Use with 3300.exe only)	Directs a parameter to the existing instance you specify. For example: `3300.exe -use:3000001` Does not create a new instance. Do not use if you are using `-new` in the same command.	Browser MMS MIDP
`-shell` (Use with `3300.exe` only))	Creates a shell in which you can repeatedly load commands to the SDK instance within the shell. See Working With a CLI Shell to Repeatedly Send Commands on page 75.	Browser MMS MIDP
`-qinst` (Use with 3300.exe only)	Displays the instance IDs of all running instances. Use `-qinst` as the sole option in a command line.	Browser MMS MIDP
`-Xphonenumber:<instance ID>`	Replaces the number of the instance you specify with another you specify. Enter only numeric input. For example, you can rename the instance +3300000 to 1234567 by entering: `emulator.exe -Xphonenumber:1234567`.	Browser MMS MIDP

Options for General Use

With the exception of `-version`, these options take effect only when the result of the command creates a new instance. Unless otherwise noted, use these options with `emulator.exe`.

Option	Description	Use for
`-version`	Displays version information about the SDK. This option does not create a new instance.	Browser MMS MIDP
`-Xinbox:<directory name>`	Specifies the directory that the new instance of the SDK checks for MMS files. For example: emulator.exe -Xinbox d:\MMSFiles. When you place MMS files in the directory, the SDK displays and then deletes them. The directory name can be absolute or relative to the location of emulator.exe. If the directory does not exist when you use the -Xinbox option, the SDK creates it. This function is available only through the command line interface.	MMS
`-Xontop`	Specifies that the SDK GUI should remain on top of all other open windows and must be minimized or moved to get at windows below it. Default value: off	Browser MMS MIDP
`-Xoutbox:<directory name>`	Specifies the directory that to which the SDK sends messages as MMS files. For example: `emulator.exe -Xoutbox d:\MMSSentFiles.` The pathname must be absolute. If the directory does not exist when you use the `-Xoutbox` option, the SDK creates it. This function is available only through the command line interface.	MMS
`-Xwap_gateway: <ip_address:port_number>`	Sets IP address and port number of the WAP gateway. You can use the following ports: 9200, 9201, 9203. For example, `Xwap_gatway:127.0.0.1::9201.` This option always affects the Browser, but it affects MIDP only if you specify `-Xprotocol:<WAP>.`	Browser MIDP

Options for Tracing Java Output

Trace output is displayed within the output window of an IDE or within the SDK DOS Console when you run the SDK as a standalone application.These tracing options apply only to Java output.

Option	Prints	Use for
`-Xverbose:allocation`	Memory allocation trace	MIDP
`-Xverbose:gc`	Garbage collection trace	MIDP
`-Xverbose:gcverbose`	Verbose garbage collection trace	MIDP
`-Xverbose:threading`	All threads in the system	MIDP
`-Xverbose:classverbose`	Verbose class loading trace	MIDP
`-Xverbose:verifier`	Verifier trace	MIDP
`-Xverbose:stackmaps`	Stackmap trace	MIDP
`-Xverbose:bytecodes`	Byte code trace	MIDP
`-Xverbose:method`	Method trace call	MIDP
`-Xverbose:methodcallsverbose`	Verbose method call trace	MIDP
`-Xverbose:stackchunks`	Stackchunk trace	MIDP
`-Xverbose:frames`	Frame trace	MIDP
`-Xverbose:exceptions`	Exception trace	MIDP
`-Xverbose:events`	Event trace	MIDP
`-Xverbose:monitors`	Monitor trace	MIDP
`-Xverbose:networking`	Networking trace	MIDP
`-Xverbose:` `-Xverbose:all`	Enable all Java traces	MIDP

Options for General Management of Java Output

You can use the options in this section to do general work with MIDlets.

Option	Description	Use for
`-classpath <directory>` (Use with 3300.exe only)	Specifies the directories and JAR files to be searched for classes; the classpath may include both library and application class directories and JAR packages.	MIDP
`-Xdescriptor:<JAD file>` (Use with `emulator.exe` only)	Runs an application using the specified JAD file.	MIDP
`-Xdebug` (Use with `emulator.exe` only)	Specifies that the MIDlet be run in debugging mode.	MIDP

Option	Description	Use for
-Xdevice:<device name> (Use with emulator.exe only)	Run an application on the SDK specified by the device name.	MIDP
-Xheapsize:<heapsize>	Specifies the heapsize (in bytes) available for MIDlets. The value may be specified in kilobytes, by appending a k to the number or in mega-bytes by appending an M to the number. The valid value range is 0 to 1048576 (1 MB). If the specified value is not in the valid range that is, more than 1048576 bytes) the heapsize is set to the default value. Default value: 211088 bytes (212 KB)	MIDP
-Xhttp_proxy: <proxy address or port number>	Specifies the address and port number of the HTTP proxy the HTTP protocol will use when making network requests Specify the value in the format: address:port_number. For example, 113.14.176.11:8080 or proxy.domain:8080. If the port number is not specified, the port 8080 is used. This option takes effect only when used with -Xprotocol:<networking protocol>.	MIDP
-Xjam: transient=<URL JAD_filename> (Use with emulator.exe only)	Specifies that the SDK is to load the specified MIDlet from the hard drive using the MIDlet provisioned method instead of the unprovisioned method.	MIDP
-Xprotocol:<networking protocol>	Specifies whether to use WAP or HTTP networking protocol for MIDlets.	MIDP
-Xquery (Use with emulator.exe only)	Print information about the SDK. This option is not valid when used with the -Xjam option.	MIDP
-Xrunjdwp:address=<port> (Use with emulator.exe only)	Specifies the TCP/IP port number between the emulator KVM and the debug proxy.	MIDP
-Xrunjdwp:<dt_socket> (Use with emulator.exe only)	Specifies the transport mechanism used to communicate with the debugger. Only dt_socket is supported.	MIDP
-Xrunjdwp:server=<port> (Use with emulator.exe only)	Starts the debug agent as a server. The debugger connects to the port specified in <port>.	MIDP
-Xrunjdwp:suspend=<y> (Use with emulator.exe only)	Suspends the VM immediately after establishing a connection to the debugger. Only the value Y is supported.	MIDP

About Parameters

The following table lists the parameters you can use on the command line when you use `3300.exe`:

Command Line Parameters	Description	Values/Options
`load <value>` (Use with `3300.exe` only)	Specifies a filename or a URL string to be loaded into an SDK instance. For example: • To load a local WML file: `3300.exe load C:\welcome.wml` • To load a remote WML file: `3300.exe load http://server.domain/welcome.wml` You can imply the load parameter when you are not working within a shell (see Working With a CLI Shell to Repeatedly Send Commands on page 75): `3300.exe http://server.domain\welcome.wml`	• MMS filename • WML filename • URL string • JAD filename • JAR filename
`close` (Use with `3300.exe` only)	The close parameter works in two ways, depending whether you are within a shell or outside a shell (see Working With a CLI Shell to Repeatedly Send Commands on page 75): • When you use `close` within a shell, `close` terminates the shell. To also terminate the instance, you must use close with `-exit`. The parameter `close` terminates the shell and the option `-exit` terminates the instance associated with the shell. • When you use `close` outside a shell, the parameter does nothing unless you use it with either `-exit` (to terminate the highest numbered SDK instance or the instance you specify with `-use`) or `-exitall` (to terminate all open SDK instances). Unlike other options, `-exit` and `-exitall` work only with the `close` parameter and, when used, must immediately follow the parameter.	`-exit`. Closes one of the following: • Within a shell: the SDK instance within the shell. • Outside a shell: the SDK instance you specify with -use or the SDK instance with the highest number. `-exitall`. Used only with the `close` parameter outside a shell to close all running SDK instance.

You can enter only one parameter on the command line.The parameter must be the last item on the line after any options on the line. For example:

- `3300.exe load http://server.domain/welcome.wml`. This example loads the site `http://server.domain/welcome.wml` into the SDK instance that is created.
- `3300.exe close -exitall`. This example closes all open Nokia 3300 SDK instances.

Working With a CLI Shell to Repeatedly Send Commands

Use the `-shell` option with `3300.exe` to create a session where you can repeatedly send commands to the one instance without having to recreate the instance with every parameter. You can attach a shell to an instance that was created through a standalone SDK or through an application such as JBuilder. In this example, more than one instance is running:

```
3300.exe -shell -use:330001
%%-->> instance : 330001

->  load http://server.domain/welcome.wml
%%-->> load : ok

->  load c:\welcome.wml
%%-->> load : ok

->  close

%%-->> detaching
```

Create a shell using instance 3300001.

Load this web site into instance 3300001 in the shell.

Load this `WML` file into instance 3300001 in the shell.

Close the shell. Instance 3300001 continues to exist.

To close both the shell and the instance within the shell, use the `close` parameter with the `-exit` option:

```
-> close -exit
%%-->> exiting
```

In the example above, the `close` parameter closes the shell session, and the `-exit` option closes the instance within the session.

Using Inboxes and Outboxes With MMS Messages

Using the command line interface, you can set up a situation where the SDK can automatically:

- Retrieve and display MMS message files from a directory you specify with the `-Xinbox` option
- Send MMS message files to a directory you specify with the `-Xoutbox` option

You can take advantage of `-Xinbox` option if you have set up a deployment directory in an IDE into which the MMS messages you create are automatically saved as MMS files, but you can also manually place a message file into the deployment directory. In either arrangement, the SDK automatically retrieves the MMS message files from that directory, displays the message, and deletes the file.

With the `-Xinbox` and `-Xoutbox` options you can set up a situation in which two SDKs instances share inboxes and outboxes so that you can transfer messages from one SDK instance to another. You may want to do this test to see whether a MIDI file can be forwarded within an MMS message. (You should not be able to forward a DRM forward-locked protected file.) Or you can set up this situation to test how one MMS message appears on two different SDKs (for example, the 3300 and the 3510i SDKs).

To set up a situation where two SDKs share inboxes and outboxes:

1 Create an SDK instance 3300000 that automatically retrieves message files from `box_a` and saves the sent message files to `box_b` by typing: `3300.exe -Xinbox c:\box_a -Xoutbox c:\box_b`

2 Create SDK instance 3300001 that automatically retrieves message files from `box_b` and saves the sent message files to `box_a` by typing: `3300.exe -Xinbox c:\box_b -Xoutbox c:\box_a`

3 Create a message and save it to `c:\box_a`.

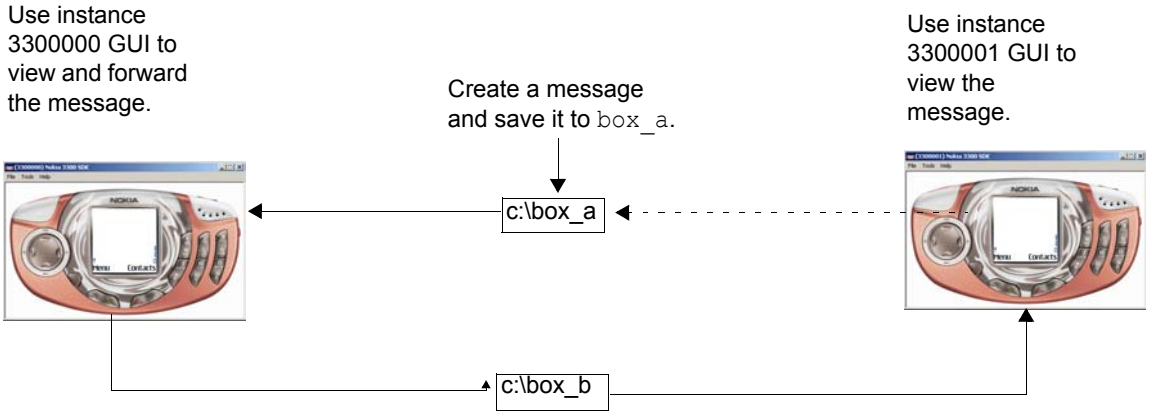
4 SDK instance 3300000)automatically retrieves the message file from box_a and deletes the file from `box_a`.

5 Use the GUI on SDK instance 3300000 to view and forward the message.

6 SDK instance 3300001 picks up the message from `box_b`.

7 Use the GUI on SDK instance 3300001 to view. the message. Optionally, you can forward it to box_a.

Conceptually, the flow of the message looks like this:


Use instance 3300000 GUI to view and forward the message.
Create a message and save it to box_a.
Use instance 3300001 GUI to view the message.
c:\box_a
c:\box_b

If a message cannot be forwarded because it contains a protected format such as MIDI, SDK 3300000 posts an announcement stating that it cannot send the message.

You can create a batch file to set up this situation. The batch file would look something like this:

```
@echo off
3300.exe -Xinbox:c:\box_a -Xoutbox:c:\box_b
3300.exe -Xoutbox:c:\box_b -Xoutbox:c:\box_a
```

NOKIA
CONNECTING PEOPLE

NOKIA
CONNECTING PEOPLE